TU Clausthal

Faster and More Robust Algorithms for Monte Carlo Light Transport Simulation

Doctoral Thesis
(Dissertation)

submitted to the department of informatics of Clausthal University of Technology in partial fulfillment of the requirements for the degree of

Doctor Rerum Naturalium (Dr. rer. nat.)

Submitted by
Johannes Jendersie

Clausthal-Zellerfeld, December 2019

Date of oral examination
26 June 2020

Faster and More Robust Algorithms for Monte Carlo Light Transport Simulation

 Johannes Jendersie  0000-0003-0703-4433
Place of birth: Berlin (Germany)

Reviewer

Prof. Dr. Thorsten Grosch, TU Clausthal, Germany
Prof. Dr.-Ing. Carsten Dachsbacher, Karlsruhe Institute of Technology, Germany

Chairperson of the Board of Examiners

Prof. Dr. Sven Hartmann. TU Clausthal, Germany

Dean

Prof. Dr.-Ing. Volker Wesling, TU Clausthal, Germany

Abstract

While rendering is a well established research topic, the demands for correct and fast light transport simulations still pose open challenges. Dependent on material and scene configurations, the results of modern Monte Carlo methods can be quite noisy and even miss important effects. Therefore, both the correctness and the speed depend on the improvement of sampling algorithms.

In this thesis I propose several modifications which improve the reliability of transport methods for difficult situations. The first problem solved is the improved weighting when combining multiple samplers. Current algorithms, based on photon mapping, tend to overestimate the importance of single techniques if parts of the results are reused for different transport paths.

Another open problem is the rendering of caustic effects dependent on the scene and the selection of sampling techniques. Here, I explore two different solutions to reduce the variance in these situations. The first changes the materials locally to reduce the noise in general. This leads to blurry results which can be partially compensated by applying adaptive heuristics. The other solution is a new transport operator which makes use of the connections toward light sources to partially guide the photon transport to important regions. This improves the sampling of caustics with far distant light sources.

To achieve the described solutions I developed several useful data structures which might apply to other problems. Two of them – a hash grid and an octree – are targeted for the density estimation of particles in massive parallel algorithms. Finally, I experimented with a cheap footprint estimate as an alternative approach to calculate the density of particles in a target region.

Zusammenfassung

Im etablierten Forschungsfeld des Renderings stellen die Ansprüche an Geschwindigkeit und Korrektheit noch immer Herausforderungen. Abhängig von Materialien und anderen Szeneneigenschaften sind die Ergebnisse moderner Monte Carlo Simulationsverfahren verrauscht oder gar nicht in der Lage bestimmte Lichteffekte zu reproduzieren. Daher hängen sowohl die Korrektheit als auch die Geschwindigkeit von der Verbesserung der Sampling-Algorithmen ab.

In dieser Dissertation schlage ich mehrere Modifikationen vor, welche die Zuverlässigkeit solcher Simulationsmethoden in schwierigen Situationen erhöhen. Das erste Problem, für welches ich Lösungen präsentiere, ist die verbesserte Kombination mehrerer Sampling-Techniken bei Verwendung von *Photon Mapping*. Aktuelle Algorithmen überschätzen die Wichtigkeit bei der Wiederverwendung von Photonen, was in einer erhöhten Varianz resultiert.

Ein anderes offenes Problem ist das Rendering von Kaustiken in bestimmten Szenen oder unter Verwendung ausgesuchter Sampling-Techniken. Hierfür präsentiere ich zwei Ansätze, die die Varianz in den entsprechenden Situationen reduzieren können. Der erste ist eine lokale Anpassung von Materialien mit dem Ziel generell weniger Varianz bei beliebigen Samplern zu erzwingen. Dies führt zu einer fehlerhaften Weichzeichnung von Glanzeffekten, was wiederum durch den Einsatz von adaptiven Heuristiken verringert werden kann. Der zweite Ansatz zeigt eine neue Sampling-Technik, welche die Verbindung zu einer Lichtquelle ausnutzt um den Transport von Photonen in die sichtbaren Regionen zu lenken. Dies verbessert das Sampling von Kaustiken von weit entfernten Lichtquellen erheblich.

Des weiteren habe ich zur Umsetzung der genannten Verbesserungen mehrere Datenstrukturen entwickelt, welche auch in anderen Anwendungen Verwendung finden könnten. Zwei der Datenstrukturen – ein Hash-Gitter und ein Octree – sind darauf spezialisiert die Dichte von Partikeln an beliebigen Punkten abzuschätzen. Beide sind für den Einsatz in hoch parallelen Architekturen entworfen. Zur alternativen Schätzung von Dichten habe ich mich in dieser Dissertation außerdem mit der Abschätzung von Sampledichten aus dem Pfad selbst (*Footprints*) auseinander gesetzt.

Acknowledgements

I want to thank my supervisor Thorsten Grosch for giving me a lot of freedom in choosing research projects to pursue. Especially in the later years, this allowed me to dig into the most promising discoveries instead of following a fixed scheme. Further, Thorsten always had an open ear for any kind of problem to practically every time.

Likewise, my thanks go to my colleagues Kai Rohmer, Feng Gu, Florian Bethe and Felix Brüll for all the nice discussions and joined projects. Some of the discussions took hours of their time just to find out if an idea is worth closer research.

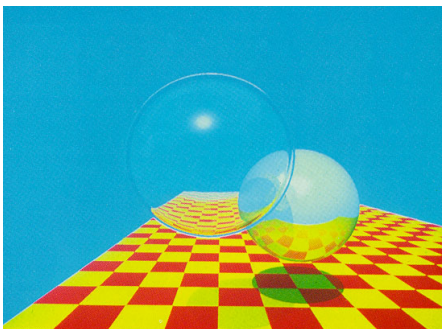
Finally, I want to thank David Kuri for our two joint publications which based on his Master Thesis. He let me have the first authorship, although we both contributed similar much work and ideas.

Special thanks go to the German Research Foundation (DFG) for founding parts of my research projects (Grant Nr. GR 3833/3-1).

Contents

Chapter I
Introduction

The rendering of photo-realistic scenes is a goal since the beginning of computer science. In the 1980s the foundations of the still used light transport techniques were developed. Parts of the widely adopted scene descriptions, like materials, even go back to the 1960s. Over the years two things changed: While the computational power increased continually, the complexity of materials and scenes grew at the same rate. Rendering an image today takes the same time as in the beginnings of computer graphics, but the degree of realism, and with it the challenges, was altered a lot.

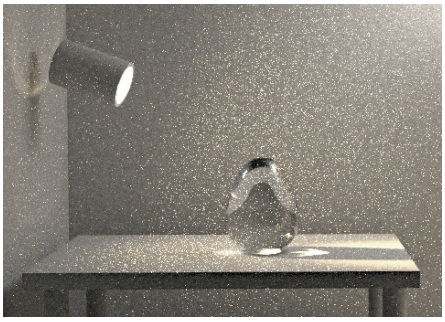


First ray-traced image [Whitted 1979]

San Miguel ([PBRT-v2 2010], Model by Guillermo M. Leal Llaguno)

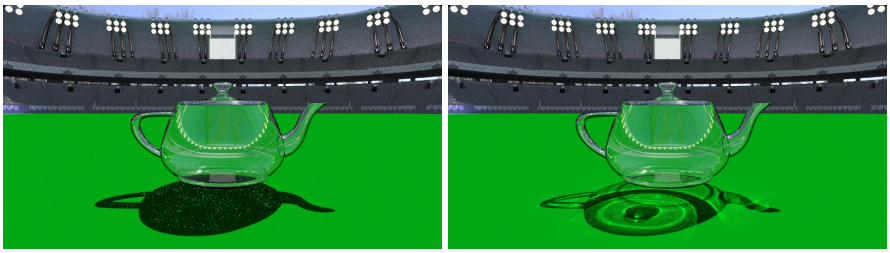
Moana island scene (Courtesy of [Walt Disney Animation Studios 2018])

Figure I.1: The change of scene complexity over time.



(a) Unexpected high noise in VCM (Chapter IV)

(b) Glossy and specular paths in light-weight methods (Chapter V, VII)



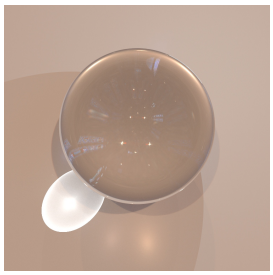
(c) Distant caustics (Chapter VI)

Figure I.2: Open problems in rendering for which this thesis proposes solutions.

For my PhD thesis I chose to improve the robustness of light transport methods. Even simple scenes can break the state of the art rendering algorithms if containing selected illumination and material setups.

After implementing one of the most robust light transport methods currently available, namely *Vertex Connection and Merging* (VCM) [Georgiev et al. 2012; Hachisuka et al. 2012], I was surprised to find an even higher variance than in simpler methods in the first scene I rendered (see Figure I.2). It turned out that VCM has a fundamental underestimation of the variance caused by parts of the sampler. I developed two different solutions which both increase the robustness considerably while still not being optimal. Nevertheless, I consider this contribution the most important within this thesis (Chapter IV).

Caustic



SDS

Other challenging problems occur if specular or glossy materials are used. In such scenes caustics, specular-diffuse-specular (SDS) paths and highly glossy paths produce a very high variance. A caustic appears if light is reflected by very smooth surfaces before hitting a rough surface. Similarly, an SDS path could be defined as a reflected caustic, for example seen through a glass or water surface. Both become difficult if the applied algorithm cannot sample this kind of effect, which is often the case in light-weight production renderers. The common handling of this problem is often the removal of the respective effect by clamping high variance contributions. This demands for new solutions which are able to reintroduce the lost light paths at a small overhead. In the chapters V and VII two different approaches are explored.

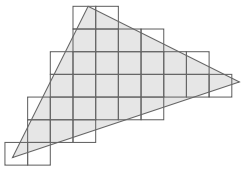
Even if the algorithm is capable of sampling caustics, it may still happen that it fails. For example a tiny caustic generating object in a large scene, like a teapot in a stadium (Figure I.2 (c)), may be explored insufficiently. A solution working for some of these situations is proposed in Chapter VI.

I.1 RENDERING APPROACHES

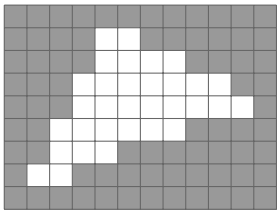
Before going into details, this section introduces the basic rendering concepts. There are two fundamental operations, rasterization and ray tracing, to generate images with a computer. Today, ray tracing is used in all high quality rendering applications and is starting to be used in real-time scenarios, too. For years, ray tracing was too slow, however advances in acceleration structures and increased computational power make it possible to trace more than 1 Grays per second. With NVIDIA's Turing series' hardware support and the introduction of ray tracing to consumer APIs like Vulkan this operation is becoming more popular than ever before.

I.1.1 RASTERIZATION

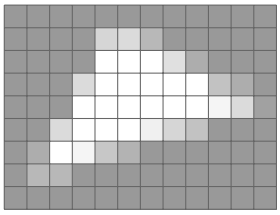
Rasterization of a triangle



Aliased result



Anti-aliased result



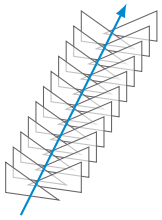
Real-time and interactive rendering algorithms are mainly based on rasterization pipelines. In rasterization primitives (usually triangles) are transformed linearly first, before being converted into pixels in screen space. Usually, a pixel is generated if its center is within the transformed primitives. Each pixel is then shaded with a programmable pipeline step (the *Shader*). It is possible that multiple fragments are generated and shaded for a single pixel to overcome aliasing artifacts (jagged object boundaries).

The rendering time with rasterization depends linearly on the number of primitives. Even if geometry is occluded or too small to generate a fragment, it is transformed first. One of the most effective optimizations here is to discard non-visible geometry by frustum culling (outside visible region) and occlusion culling (behind other visible geometry).

Effects which change the direction of light rays locally are not directly available and require multiple passes if they are possible at all. There are techniques to approximate shadows, reflections and refractions. All require additional passes which render the scene from a different point of view. Effects like caustics are even more difficult and require a splatting of sparsely sampled photons, which are either ray-traced or generated by a multi-pass rasterization.

I.1.2 RAY TRACING

Worst case for ray tracing performance



In ray tracing intersections between rays and primitives are searched explicitly. Each ray can have an individual origin or direction. To render an image similar to rasterization, one ray must be shot through the center of each pixel. Naturally, not every ray is intersected against every primitive of the scene. Instead Bounding Volume Hierarchies (sBVHs) [Aila and Laine 2009; Karras 2012; Stich et al. 2009] are used to prune the search space effectively. On average, this leads to a logarithmic scaling of the rendering time with the scene complexity. Note that it is possible to construct scenes for which the ray intersection time is still linear as depicted on the right.

Most light effects are much simpler with ray tracing. Finding a shadow, for example, only requires an any-hit ray test between the receiving surface and the light emitter. If there is any intersection inside this ray interval, the surface is shadowed. However, a full solution of the light transport problem,

including multiple scattered light, requires more complex algorithms based on any of the two basic operations.

I.1.3 MONTE CARLO RENDERING

The entire light transport problem can be described by a recursive integral equation which will be introduced in Section II.7.1. In short, the illumination of a point depends on the integral over all incoming light. The light which is scattered by this point is then illuminating all other points in turn. Our searched solution is the equilibrium state of this system.

To solve the integral we apply *Monte Carlo* (MC) integration. Therefore, random light paths are generated to sample the space of all possible transport paths. The expected value of this random experiment is the desired solution. The nature of this random process is that each estimate will have variance, causing a noisy output image. By iteratively repeating the sampling, the solution converges over time. Dependent on the complexity of the scene this takes minutes to hours and can even exceed an entire lifetime. Still, it is the best known way for general purpose light transport and the ongoing research focuses on faster convergence rates.

MC Sec. II.2.2 p. 20

To reach faster convergence, this thesis proposes several solutions to reduce the variance of otherwise worst case configurations. The general goal being an equally fast convergence of all possible effects.

I.1.4 PRODUCTION RENDERING

Production rendering is a term commonly used for applied renderers opposed to research renderers. It is used in movie production, architectural design and digital product advertisement. A production renderer must reach convergence within a feasible time, often at the cost of discarding hard to sample effects.

Besides that it has to face two further challenges: scalability and artists. Often movie scenes have huge dimensions, barely fitting into a main memory with 128 GB. An example is the MOANA ISLAND scene shown in Figure I.1. It consists of 95 M primitives which are instanced 28 million times which totals in 15 G primitives in the full rendering. On the other hand, the artists require a large degree of freedom which, at times, diverges from the physically correct solution. To allow both – huge scenes and a high degree of freedom – only simple MC methods are applied. Basically all production renderers are based on *Path Tracing* (PT) [Bala 2018; Burley et al. 2018; P. Christensen et al. 2018; Fascione et al. 2017a; Georgiev et al. 2018] which is the simplest MC method with regard to the sampling space. Without modifications PT is not able to sample caustic or SDS paths.

I.2 PROBLEM STATEMENT

In this section the problems noted previously are summarized to focus on what this thesis is going to improve. Monte Carlo rendering is a well explored research field dating back to Kajiya [1986]. There is a large foundation for the description of scenes, materials and specific light effects. Nevertheless, some problems are not solved satisfyingly.

PROBLEM I: Reliable low variance combination of rendering methods.

It is a known fact that Veach's balance heuristic [Veach and Guibas 1995b] does not weight the different techniques in a variance optimal way. A recently presented solution [Kondapaneni et al. 2019] is not yet included in this thesis. However, even the optimal weights from Kondapaneni et al. [2019] will suffer from the same problems as the balance heuristic, when it comes to a partial reuse of the sampler. Thus, the weights for the optimal combination of rendering techniques remains an open problem.

PROBLEM II: Limited variance independent of the material configuration.

Specular and glossy paths still cause high variance configurations in the state of the art rendering methods. Since rendering time is mainly governed by the most noisy light effects, it is very important to improve the samplers for these configurations.

PROBLEM III: Reduced memory and computational overhead.

In production rendering the simpler sampling methods are preferred for several reasons. To become adopted in practical applications, PROBLEM I and II should be achieved with as little overhead as possible. If possible, the same results should be available with several rendering methods.

I.3 List of Contributions

Most contributions in this thesis were published in peer-reviewed formats previously. Six of the following papers are directly incorporated into Chapters III to VII, although I restructured the presentation opposed to the papers. The shared content is only introduced once and cross referenced where applicable.

[Jendersie et al. 2016a] *Precomputed Illuminance Composition for Real-time Global Illumination*
Johannes Jendersie, David Kuri and Thorsten Grosch
In: Proc. of SIGGRAPH Symposium on Interactive 3D Graphics and Games

[Jendersie et al. 2016b] *Real-Time Global Illumination Using Precomputed Illuminance Composition with Chrominance Compression*
Johannes Jendersie, David Kuri and Thorsten Grosch
In: Journal of Computer Graphics Techniques

[Jendersie et al. 2017] *Pixel Cache Light Tracing*
Johannes Jendersie, Kai Rohmer, Felix Brüll and Thorsten Grosch
In: Proc. of Vision, Modeling and Visualization

[Rohmer et al. 2017] *Natural Environment Illumination: Coherent Interactive Augmented Reality for Mobile and non-Mobile Devices*
Kai Rohmer, *Johannes Jendersie* and Thorsten Grosch
In: IEEE Transactions on Visualization and Computer Graphics (Proc. of ISMAR)

[Gu et al. 2018] *Fast and Dynamic Construction of Bounding Volume Hierarchies based on Loose Octrees*
Feng Gu, *Johannes Jendersie*, Thorsten Grosch
In: Proc. of Vision, Modeling and Visualization

[Jendersie 2018] *Path Throughput Importance Weights*
Johannes Jendersie
In: arXiv:1806.01005

[Jendersie and Grosch 2018] *An Improved Multiple Importance Sampling Heuristic for Density Estimates in Light Transport Simulation*
Johannes Jendersie and Thorsten Grosch
In: Proc. of Eurographics Symposium on Rendering EI&I Track

[Jendersie 2019b] *Variance Reduction via Footprint Estimation in the Presence of Path Reuse*
Johannes Jendersie
In: Ray Tracing Gems 1st ed., Edited by Eric Haines and Tomas Akenine-Möller

[Jendersie and Grosch 2019] *Microfacet Model Regularization for Robust Light Transport*
Johannes Jendersie and Thorsten Grosch
In: Computer Graphics Forum (Proc. of EGSR)

[Jendersie 2019a] *Next Event Backtracking*
Johannes Jendersie
In: arXiv:1909:00573

The first two publications in the list are not part of this thesis, because they are not related to the improvement of MC algorithms. Instead, the two papers were targeted to interactive and real-time global illumination. They are based on the master thesis of David Kuri which I supervised. By using a surfel-based hierarchy to represent the scene, we were able to precom-

pute and cluster the most important light paths. This enabled the indirect illumination from dynamic light sources onto static and dynamic geometry at interactive rates, including soft shadows for indirect lighting and area lights. Due to an iterative processing the simulation even finds multiple bounces over time. In the second version we used an alternative color space (YCoCg [Malvar and Sullivan 2003]) to reduce the memory consumption of our method. A successor of this method by Silvennoinen and Lehtinen [2017] improves the shadow quality while requiring less memory than ours.

The next publication which is not part of this thesis is the *Natural Environment Illumination* [Rohmer et al. 2017]. It describes a full system which achieves interactive global illumination in augmented reality on a single tablet device. The most important contribution is the capturing and registration of a high dynamic range environment. I mainly contributed to the rendering process which is based on a sparse Monte Carlo sampling. We experimented with several types of cone casts to capture integrated shadow and irradiance information from the recorded environment.

Finally, there is the acceleration structure paper [Gu et al. 2018] which targeted ray tracing in particle simulations. We improved the very fast *Linear BVH* (LBVH) method by Karras [2012] in the highly dynamic context of particle mixture simulations. Besides frequent changes, this kind of simulation favors millions of heterogeneously sized particles with high occlusions. For this data, LBVH creates low quality BVHs leading to a higher round-trip-time composed of building the acceleration structure and tracing. The quality can be improved by assigning geometry to different levels of the hierarchy. Indeed, our results were better than LBVH for the targeted scenario. However, we also applied the construction method to conventional triangle-mesh scenes which did not yield significant improvements. Therefore, it is of low importance for this thesis.

I.4 OUTLINE

In Chapter II the basic concepts of Monte Carlo rendering and parts of the physical background are introduced. If you are familiar with the topic you may skip this chapter. I will make heavy use of the side margin to refer back to symbols and formulas such that it should be easy to find the meaning of specific notations upon encounter. Almost anything within this chapter, I learned after finishing my master degree which explains the chosen level of detail. So, a basic understanding of computer graphics is assumed.

Besides the common basics, there are also minor improvements and contributions in the fundamentals chapter. Most noticeable the alternative uniform sampling of triangles and the importance sampling of the Oren-Nayar material model [Oren and Nayar 1994].

Uniform triangle sampling Sec. II.3.2 p. 30
Oren-Nayar importance sampling Sec. II.5.4 p. 45

In several publications I required fast ways to assess the local particle density [Jendersie 2019a; Jendersie and Grosch 2018]. To that purpose, I developed dedicated data structures which are presented in Chapter III. None of the papers explored the proposed data structures in the depth presented here.

Chapter IV then presents two solutions to PROBLEM I based on the published articles [Jendersie 2019b; Jendersie and Grosch 2018]. Both approximate the effective benefit of reusing photons in VCM leading to a more robust rendering method.

PROBLEM I to III Sec. I.2 p. 5

The Chapters V to VII present three different approaches to tackle PROBLEM II and III. The microfacet regularization (Chapter V, [Jendersie and Grosch 2019]) is able to introduce all lighting effects into any kind of renderer, although results are biased and noisy. *Next Event Backtracking* (Chapter VI, [Jendersie 2019a]) produces higher quality photon maps based on implicit guidance for the first segment of a light path. Finally, *Pixel Cache Light Tracing* (Chapter VII, [Jendersie et al. 2017]) shows an attempt for a very light-weight rendering approach. Due to my inexperience back in 2017, I made several decisions which sacrificed advantages of other transport methods. Therefore, in Chapter VII I included discussions on the lessons learned and potential improvements.

The last Chapter VIII summarizes the contributions of this thesis.

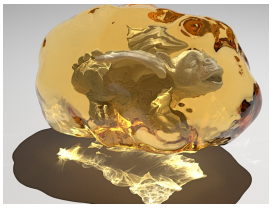
I.5 SYMBOLS AND FUNCTIONS

The following table gives an overview of conventions and functions used in all formulas. It does not introduce the specific sizes and symbols of the used quantities which will be introduced when needed.

Example	Definition
f, F	A function, lower and upper case are used dependent on context
p	Probability Density Function (PDF) (p used exclusively)
P	Probability (P used exclusively)
t	Throughput f/p in Monte Carlo estimators
$\boldsymbol{d}$	Vectors (*lower case, bold*). For low dimensional vectors $.x$, $.y$ and $.z$ are used to select a single component.
$\boldsymbol{J}$	Matrices (*upper case, bold*).
$\mathcal{X}$	A set (*upper case, calligraphic*).
$[\square, \square]$	Closed interval
$[\square, \square)$ or $(\square, \square]$	Half open interval, open interval $(\square, \square)$
$(\square)^+$	*Positive part* function $\max(0, \square)$
$()^T$	Transpose matrix or vector
$\lvert\square\rvert$	Absolute value
$\lVert\square\rVert$	Euclidean norm for vectors and absolute determinant for matrices
$\langle\square, \square\rangle$	Scalar (dot) product of vectors
$\square_\downarrow$	The size depends on the incident direction which is a fixed direction of some path. It is *not* the direction of the light transport.
$\square_\uparrow$	The size depends on the excitant direction which is the outgoing direction in sampling and evaluation events. Again, this is not the direction of a specific quantity.

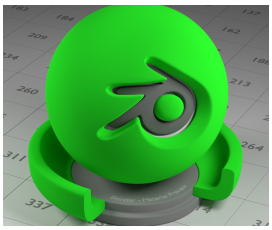
I.6 List of Scenes

I really acknowledge those who publish high quality content for research and other uses. To give appropriate credits, the following list introduces the scenes used in my experiments. The list is ordered alphabetically with respect to the used names within this thesis. All scenes not listed here are self-made.



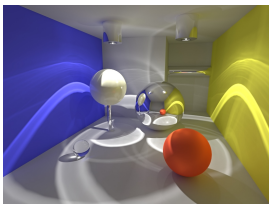
AMBER
Dragon from Christian Schüller, taken from the PBRT-v3 repository
https://www.pbrt.org/scenes-v3.html
Modified: Added amber, Tessellation

BATHROOM (Contemporary-bathroom)
Courtesy of Mareck (CC-Zero), taken from the PBRT-v3 repository
https://www.pbrt.org/scenes-v3.html
Modified: Manifolds, Materials



BLENDER PROBE (Cycles Material Test)
Courtesy of Robin Marin (CC-BY-SA), modified by Gottfried Hofmann
https://www.blenderdiplom.com/en/downloads/584-download-cycles-material-test-scene.html
Modified: Manifold / Tessellation quality

CHRISTMAS
Courtesy of Deathtome (CC-BY), taken from
https://www.blendswap.com/blend/view/63719
Modified: Materials, removed curves



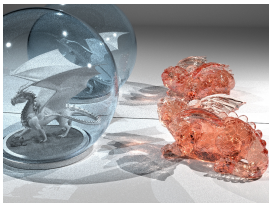
MIRRORBALLS
Scene from Toshia Hachisuka, taken from
https://github.com/PetrVevoda/smallupbp/tree/master/scenes/mirrorballs
Modified: Materials

NEB bias test
LUCY and ASIAN DRAGON from Stanford scanning repository
http://graphics.stanford.edu/data/3Dscanrep/
Modified: Manifolds, surrounding scene

NECKLACE (Evenstar Necklace)
Courtesy of pizzahouse6 (CC-BY), taken from
http://www.blendswap.com/blends/view/56411
Modified: Materials

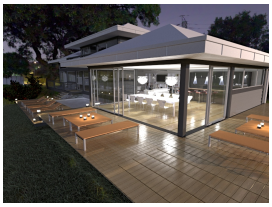
SPONZA
Crytek version (originial from Marko Dabrovic), taken from https://casual-effects.com/data/ with thanks to Morgan McGuire
Modified: Materials

TEAPOT IN A STADIUM
Utah Teapot, Stadium from Ericchip1983 (CC-BY), taken from
https://www.blendswap.com/blend/view/74526
Modified: Composition



TOY DRAGONS
PBRT Dragon and a second dragon from Delatronic (CC-BY), taken from
https://www.blendswap.com/blend/view/80766
Modified: Added surrounding geometry

VEACH-BIDIR
Scene from Eric Veach, taken from the PBRT-v3 repository https://www.pbrt.org/scenes-v3.html
Modified: Tessellation, Materials



VILLA
Courtesy of Florent Boyer, taken from the PBRT-v3 repository https://www.pbrt.org/scenes-v3.html
Modified: Tessellation, Materials

WATCH
Courtesy of heraSK (CC-BY), taken from
http://www.blendswap.com/blends/view/70232
Modified: Manifolds, Materials

Chapter II
Fundamentals in Monte Carlo Light Transport Simulation

This chapter lays the fundamental concepts and equations required for all upcoming chapters. It is not meant as a complete introduction to the mathematics and models of computer graphics. The reader is referred to the PBRT Book [Pharr et al. 2017] for more details. Also, applied algorithms, related works and details are introduced if needed later on.

As in most other works in computer graphics, light propagation is described by *geometric optics* rather than *wave optics* or *quantum optics*. In *geometric optics* all processes are described by a particle model for which a particle travels along a ray until an interaction with matter occurs. Neither the travel time nor effects like interference are modeled in the following. This also includes that an image is always a snapshot of one point in time, while still showing the equilibrium stage of emitted light. This implies that light travels infinitely fast, which is a reasonable assumption for the models in computer graphics.

In common a light particle is called a *photon*. It transports a small amount of *radiant energy* and must somehow be registered on a camera sensor to contribute to the image. Its *adjoint*, the particles sent from the camera, are called *importons*. Their density describes the importance, that is the influence of a region in the scene to the final image. Both quantities are treated symmetrically for most of the time.

II.1 Radiometric Quantities

Radiometric quantities describe the light propagation, starting from pure energy to the measurable brightness of a surface. They are defined with respect to a wavelength λ. Also, there is an equivalent family of *photometric quantities* which describe the perceived color and brightness of the light to the human eye.

To convert from the *radiometric quantity* into the photometric one, a weighted integral over all (visible) wavelengths must be taken. As a weight the sensory response curves $V(\lambda)$ of the human receptors in the eye are applied. Since most humans have a trichromatic vision we use the Red-Green-Blue (RGB) color model in computer graphics. In 1931 the CIE 2° standard

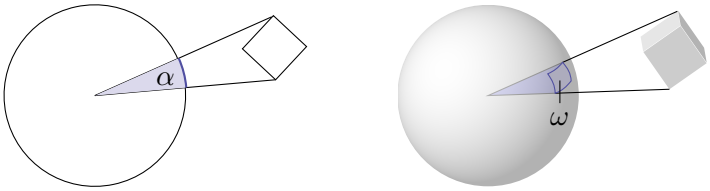


Figure II.1: Comparison between planar angle α and solid angle ω. A planar angle measures the arc length of an object projected to a unit circle. A solid angle measures a surface patch of a projection to the unit sphere.

observer model was introduced, which defines the response curves V for a $2°$ field of view. The tabulated values of these curves can be found in [Guild et al. 1931]. Since color vision changes with the field of view, a $10°$ standard observer was introduced by CIE in 1964, but is rarely used.

The transport algorithms in this dissertation can be applied to both kinds of quantities. Without loss of generality only the *radiometric quantities* are used. However, the notion of λ is omitted for brevity. In practical implementations there are at least three possible forms:

1. RGB-Tuple. Every parameter, material interaction and so forth is using the photometric value for RGB directly. This is the most common model, which works well for most scenarios. However, spectral effects like *dispersion*, *fluorescence* or *metamerism* cannot be captured well.
2. Single Wavelength. A random wavelength is assigned to each photon and converted to an RGB response on output.
3. Wavelength-Tuple. Larger tuples with more than the three RGB values are used. They represent samples of wavelengths and the final output value is generated by a weighted sum over these samples using $V(\lambda)$.

The single wavelength method is a straight forward solution and yields a physically correct spectral renderer. However, it is relatively inefficient as it exhibits a lot of color noise. An attempt to solve this problem is to use multiple wavelength samples (method three) on a single path as in [Evans and McCool 1999]. The state of the art solution is *Hero Wavelength* sampling [Wilkie et al. 2014] where the other wavelengths of a tuple are chosen deterministically with respect to the hero wavelength. Further, all wavelengths inside a tuple are weighted by a proper *Multiple Importance Sampling* (MIS) to optimally combine the samples which have different hero wavelength. All images in this document are rendered using the first approach for simplicity.

MIS Sec. II.2.3 p. 22

II.1.1 Solid Angles

Before it is possible to describe all the light quantities, we need to understand the concept of solid angles. A solid angle is, in general, the two dimensional equivalent of an angle in *radian*. It is measured as a fragment of the unit sphere with the unit *steradian* [sr] and may become as large as 4π sr, which is the total surface area of the sphere. Figure II.1 visualizes the analogy between an angle and a solid angle ω.

Def. Solid angle ω

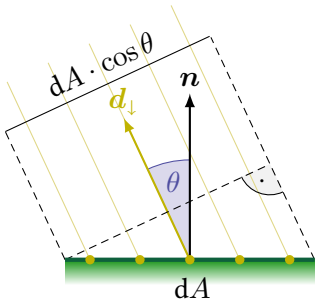


Figure II.2: Visible area. The density of photons decreases with a growing angle θ between the normal $\boldsymbol{n}$ and the incident light direction $\boldsymbol{d}_\downarrow$.

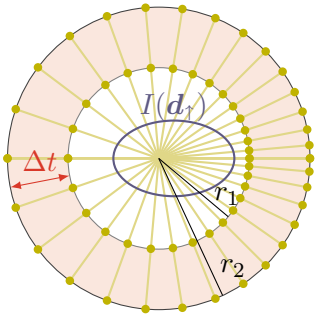


Figure II.3: Photons are distributed according to the intensity $I(\boldsymbol{d}_\uparrow)$. While propagating, their density decreases with the squared radius.

Usually, the solid angle is a measure of an amount. It does not describe a certain direction. For example $\omega = \pi\,\mathrm{sr}$ means a forth of the sphere – without any restriction to form or direction. To denote the set of directions Ω is used, where $\omega = |\Omega|$. Using a set, the *differential solid angle* $\mathrm{d}\omega$ can be associated with a certain direction $\boldsymbol{d}$. When shrinking the surface patch $\lim_{|\Omega|\to 0}$ on the unit sphere, the number of directions in Ω shrinks too. In the limit $\mathrm{d}\omega$, only one direction remains for the infinitesimal surface element. In the following integrals $\boldsymbol{d}$ is used to denote the direction which depends on the integration variable ω.

Def. $\boldsymbol{d} \Leftrightarrow \mathrm{d}\omega$

In order to compute a solid angle of an arbitrary shape, the object is projected to the sphere by integrating

$$\omega = \iint_{\mathrm{vis}(A)} \frac{|\cos\theta|}{r^2}\,\mathrm{d}A \tag{II.1}$$

Calculation of solid angle ω by projection
$|\cdot|$ is the absolute value

over its surface $\mathrm{vis}(A)$ visible from the origin of the solid angle. This restriction is necessary to exclude the backside as well as overlapping regions in concave shapes. $r = \|\boldsymbol{p}\|$ is the distance to the surface point $\boldsymbol{p} \in \mathrm{d}A$ which is given relative to the solid angle's origin. The $\cos\theta$ accounts for the visible amount of the infinitesimal surface area $\mathrm{d}A$. It can be computed by the *dot product* $\langle \boldsymbol{n}, \boldsymbol{p}/\|\boldsymbol{p}\| \rangle$ of the surface normal $\boldsymbol{n}$ and the direction towards $\boldsymbol{p}$. Also see Figure II.2 for how the cosine is connected with the visible area.

II.1.2 Total Energy and Radiant Flux

The distribution of light begins at the light source with the total amount of radiant energy Q per time unit t. This radiant power

Def. Total radiant energy Q

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t} \qquad [\mathrm{W}] \tag{II.2}$$

Def. Radiant flux Φ

is called *radiant flux*, only called *flux* henceforth. In the rendering context a *photon* is associated with a fraction of this *flux* ϕ over some time Δt. In reality a *photon* transports an amount of energy Q dependent on its wavelength. In graphics we control the number of particles and the brightness of light sources independently. Also, as already mentioned we neglect the

Def. Flux of a photon ϕ

temporal dimension. Thus, it is more useful to simulate packets of flux as particles which are also called *photon*.

Usually, the *flux* is not distributed equally among all directions. For an exitant direction $\boldsymbol{d}_\uparrow$ from a light emitting surface the *intensity*

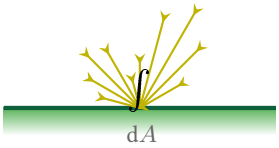
$$I(\boldsymbol{d}_\uparrow) = \frac{\mathrm{d}\Phi}{\mathrm{d}\omega} \qquad \left[\mathrm{W\,sr^{-1}}\right] \quad \text{(II.3)}$$

Def. Intensity I

describes the photon density per solid angle ω.

Figure II.3 visualizes how photon density changes according to the *intensity*. Each of the *photons* still represents a fraction of the *flux* ϕ independent of the direction $\boldsymbol{d}_\uparrow$. However, the radiant power also depends on the angular density of the *photons*.

II.1.3 Irradiance



The introduction of *flux* and *intensities* allows to describe how light is emitted from a light source. On the side of the receiver, *irradiance* describes how much flux is received per area:

$$E(\boldsymbol{x}) = \frac{\mathrm{d}\Phi}{\mathrm{d}A}. \qquad \left[\mathrm{W\,m^{-2}}\right] \quad \text{(II.4)}$$

Def. Irradiance E

It depends directly on the density of arriving *photons*.

From the previous description of the light source it is known that the particle density is proportional to I. Further, it is proportional to $1/r^2$, because the surface of the wave front increases by the squared distance of its travel. For surface light transport we also need to consider the cosine of the infinitesimal area $\mathrm{d}A$ at the target as shown by Figure II.2. A tilted surface causes a wide spread of incident photons and therefore decreases the energy density. As before, the cosine is computed by the *scalar product* $\langle \boldsymbol{n}, \boldsymbol{d} \rangle$, this time between the surface normal $\boldsymbol{n}$ and the light direction $\boldsymbol{d}$. In the following $\theta_\downarrow$ and $\theta_\uparrow$ inherit the notation of 'incident'/'excitant' from the associated direction $\boldsymbol{d}_\downarrow / \boldsymbol{d}_\uparrow$.

r: distance to the light

Taking all together, the radiant power coming from a single direction at some receiving surface point $\boldsymbol{x}$ can be described by

$$\mathrm{d}E(\boldsymbol{x}, \boldsymbol{d}_\downarrow) = \frac{\mathrm{d}I(\boldsymbol{d}_\downarrow)|\cos\theta_\downarrow|}{r^2} \qquad \left[\mathrm{W\,m^{-2}}\right] \quad \text{(II.5)}$$

Def. Differential irradiance $\mathrm{d}E$ (photometric distance law)

which is called the *differential irradiance*, also known as the *photometric distance law*. Here, the *intensity* of the light source is an incident size with respect to the surface point $\boldsymbol{x}$, meaning that light is received rather than emitted. The incident nature is denoted by the downward pointing arrow at the direction.

In consequence, the total *irradiance* received per surface point is the integral over all possible incident directions in Ω:

$$E(\boldsymbol{x}) = \int_\Omega \frac{\mathrm{d}I(\boldsymbol{d}_\downarrow) \cdot |\cos\theta_\downarrow|}{r^2} \quad \text{(II.6)}$$

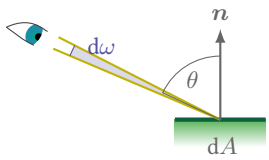
Def. Computation of irradiance E

Analogously, the total radiant flux leaving a surface (including emission and reflections), also known as *radiosity*, is defined as

$$J = \frac{\mathrm{d}\Phi_\uparrow}{\mathrm{d}A}. \quad \text{(II.7)}$$

If only the radiant flux emitted by a surface (excluding reflections and refractions) is referred to, the symbol M is used instead.

II.1.4 RADIANCE



While both $\mathrm{d}E$ and E are of importance for the computation of illumination, they do not model the observed brightness of a surface. To do so we need to introduce the *radiance*

$$L(\boldsymbol{x}, \boldsymbol{d}_\uparrow) = \frac{\mathrm{d}I(\boldsymbol{d}_\uparrow)}{\mathrm{d}A \cdot |\cos\theta_\uparrow|} = \frac{\mathrm{d}^2\Phi}{\mathrm{d}A \cdot |\cos\theta_\uparrow| \cdot \mathrm{d}\omega} \quad \left[\mathrm{W\,m^{-2}\,sr^{-1}}\right] \tag{II.8}$$

Def. Radiance L

which is the *flux* per visible area $\mathrm{d}A \cdot |\cos\theta_\uparrow|$ and excitant *solid angle*.

Note that above *radiance* is defined as an excitant size only. It also makes sense to define an incident *radiance* based on the *differential irradiance* from Equation (II.5) as

$$L_\downarrow(\boldsymbol{x}, \boldsymbol{d}_\downarrow) = \frac{\mathrm{d}E(\boldsymbol{x}, \boldsymbol{d}_\downarrow)}{|\cos\theta_\downarrow| \cdot \mathrm{d}\omega}.$$

It is the *irradiance* per *solid angle* and because of the angular dependency it is not related to the surface orientation. Referring back to Equation (II.5) we see that the $|\cos\theta_\downarrow|$ cancels out. Also, this yields an alternative formulation of the *irradiance*

$$E(\boldsymbol{x}) = \int_\Omega L_\downarrow(\boldsymbol{x}, \boldsymbol{d}_\downarrow)|\cos\theta_\downarrow|\,\mathrm{d}\omega. \tag{II.9}$$

To see that both Equation (II.6) and Equation (II.9) are the same the *radiance* from Equation (II.8) can be inserted with an inverted direction, followed by canceling the cosine terms. In the last step the *differential solid angle*, as seen from the light source, can be substituted for $d\omega$:

$$\begin{aligned} E(\boldsymbol{x}) &= \int_\Omega \frac{\mathrm{d}I(\boldsymbol{d}_\downarrow)}{\mathrm{d}A \cdot |\cos\theta_\downarrow|}|\cos\theta_\downarrow|\,\mathrm{d}\omega \\ &= \int_\Omega \frac{\mathrm{d}I(\boldsymbol{d}_\downarrow)}{\mathrm{d}A}\,\mathrm{d}\omega \\ &= \int_\Omega \frac{\mathrm{d}I(\boldsymbol{d}_\downarrow) \cdot |\cos\theta_\downarrow|}{r^2} \end{aligned}$$

From Eq. (II.8) p. 17:
$L = \mathrm{d}I \cdot \mathrm{d}A^{-1} \cdot |\cos\theta_\downarrow|^{-1}$
From Eq. (II.1) p. 15:
$\mathrm{d}\omega = \mathrm{d}A \cdot |\cos\theta_\downarrow| \cdot r^{-2}$

The first important difference between *differential irradiance* and incident *radiance* is the cosine. *Radiance* is defined with respect to the visible area, while *irradiance* is defined with respect to the total area.

The second difference is that *radiance* is invariant to the view distance because of dividing by the solid angle which also shrinks quadratically with the distance. This means that a surface is always perceived equally bright, independent of the distance.

II.2 SAMPLING

Sampling is the random choice of some quantity, given a *Probability Density Function* (PDF). It is the most fundamental requirement for Monte Carlo integration which is introduced in this section. Our overall goal is to estimate an integral

$$I = \int_\Omega f(x)\,\mathrm{d}\mu(x) \tag{II.10}$$

Def. Definite integral I

for an arbitrary function f using the measure function μ. In our case, the integral cannot be solved in a closed form. Therefore, we are going to numerically compute an *estimate* $\hat{I}$ of the integral value I by using random experiments.

Def. $\hat{I}$ numerical estimate

In this context a *sampler* is an algorithm to produce random instances, called *samples*, following a specific PDF. Drawing N *samples*, a *sampler* produces an *estimate* $\hat{I}_N$ of the searched integral. The domain of a *sampler* and its PDF may differ from context. Locally, we often sample a position on a surface or an outgoing direction. However, transport algorithms are *samplers* too, providing samples in the path domain. That means, they provide entire light transport paths, based on the sequential execution of multiple local samplers.

II.2.1 BASIC PROBABILITY AND STOCHASTIC

Before we can define the Monte Carlo sampler we need to understand the PDF. Formally, it is a non-negative function $p : \Omega \mapsto [0, \infty]$ with the property

$$\int_\Omega p(x)\,\mathrm{d}x = 1. \tag{II.11}$$

The probability to create a sample $P(X \in [a, b])$ is $\int_a^b p(x)\,\mathrm{d}x$. Equation (II.11) ensures a probability of one, that a sample is somewhere in the domain Ω.

A sampler is then performing a random experiment X whose expected value is equal to the searched integral $I = \mathrm{E}[X]$. In general, the expected value has following useful properties:

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y] \tag{II.12a}$$
$$\mathrm{E}[aX] = a\mathrm{E}[X] \tag{II.12b}$$
$$\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y] + \mathrm{Cov}[X, Y] \tag{II.12c}$$

where $\mathrm{Cov}[X, Y]$ is the covariance. It vanishes if the two variables are independent.

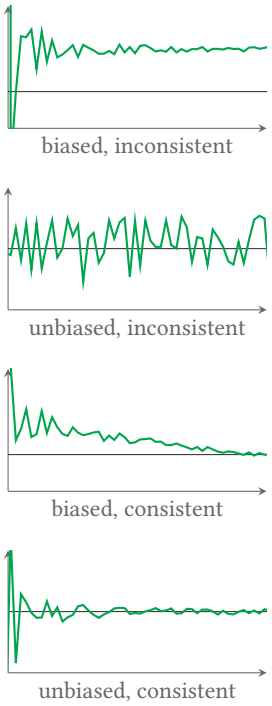
A sampler is said to be *unbiased* if the difference between the estimate $\hat{I}_N$ and the true value I is zero. The deviation from zero is the *bias*

$$\mathrm{B}[X] = \mathrm{E}\left[\hat{I}_N - I\right]. \tag{II.13}$$

A related property is to be *consistent* which guarantees the convergence to the true value

$$\lim_{N\to\infty} \hat{I}_N = I. \tag{II.14}$$

Neither of the two implies the other. It is even possible to have a *biased* but *consistent* estimator as shown in the examples at the side. For image synthesis *consistency* is the more important property. An *unbiased* estimator might even be useless if its variance is infinite. In that case it will not converge at all. An example is the sampling of caustics in a path tracer. A pure specular path has a zero probability to be sampled but a non-zero contribution causing an infinite variance.



Finally, there is the variance which tells us how far samples spread around the expected value. It is defined as the expected value of the squared deviation

$$\mathrm{V}[X] = \mathrm{E}\big[(X - \mathrm{E}[X])^2\big] = \mathrm{E}\big[X^2\big] - \mathrm{E}[X]^2 = \mathrm{E}\big[X^2\big] - I^2. \tag{II.15}$$

It has the following properties

$$\mathrm{V}[X+Y] = \mathrm{V}[X] + \mathrm{V}[Y] + 2\mathrm{Cov}[X,Y] \tag{II.16a}$$

$$\mathrm{V}[aX] = a^2\mathrm{V}[X] \tag{II.16b}$$

$$\mathrm{V}[XY] = \mathrm{E}[X]^2\,\mathrm{V}[Y] + \mathrm{E}[Y]^2\,\mathrm{V}[X] + \mathrm{V}[X]\,\mathrm{V}[Y] \quad \textit{if X,Y independent} \tag{II.16c}$$

II.2.2 Monte Carlo Integration

As mentioned in the introduction we will need to solve an integral equation which, in general, is not solvable in a closed form. Monte Carlo integration is a numerical method to estimate an arbitrary complex integral based on random samples.

Rendering Equation
Eq. (II.69) p. 63

The fundamental idea is to discretize the function into stripes and summing up their areas. Thereby, we chose the stripes randomly according to a PDF p and let their width be dependent on $1/p$. Figure II.4 visualizes this choice. In regions of high sample density each sample should be associated with a thinner stripe. Thus, the area of a stripe is $f(x)/Np(x)$ where N is the number of stripes.

Mathematically, we compute an *unbiased* estimate of the expected value of the total area

$$I = \mathrm{E}\left[\frac{f(x)}{p(x)}\right] = \int_\Omega \frac{f(x)}{p(x)} p(x)\,\mathrm{d}x \tag{II.17}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} = \hat{I}. \tag{II.18}$$

To be *unbiased* the PDF must be greater zero whenever f has a contribution ($f(x) \neq 0 \Rightarrow p(x) > 0$). Under this condition, the law of large numbers guarantees us that the estimator will converge for $N \to \infty$. Since it is *unbiased* and converging it is also *consistent*.

The term $f(x)/p(x)$ of sampling events on a light path will also be referred to as the *throughput* $t(x)$. It is a weight of how much a sample is going to contribute to the integral.

The actual speed of convergence depends on the variance

$$\mathrm{V}[I] = \mathrm{V}\left[\frac{f(x)}{p(x)}\right] = \mathrm{E}\left[\frac{f(x)^2}{p(x)^2}\right] - \mathrm{E}\left[\frac{f(x)}{p(x)}\right]^2 = \int_\Omega \frac{f(x)^2}{p(x)^2} p(x)\,\mathrm{d}x - I^2. \tag{II.19}$$

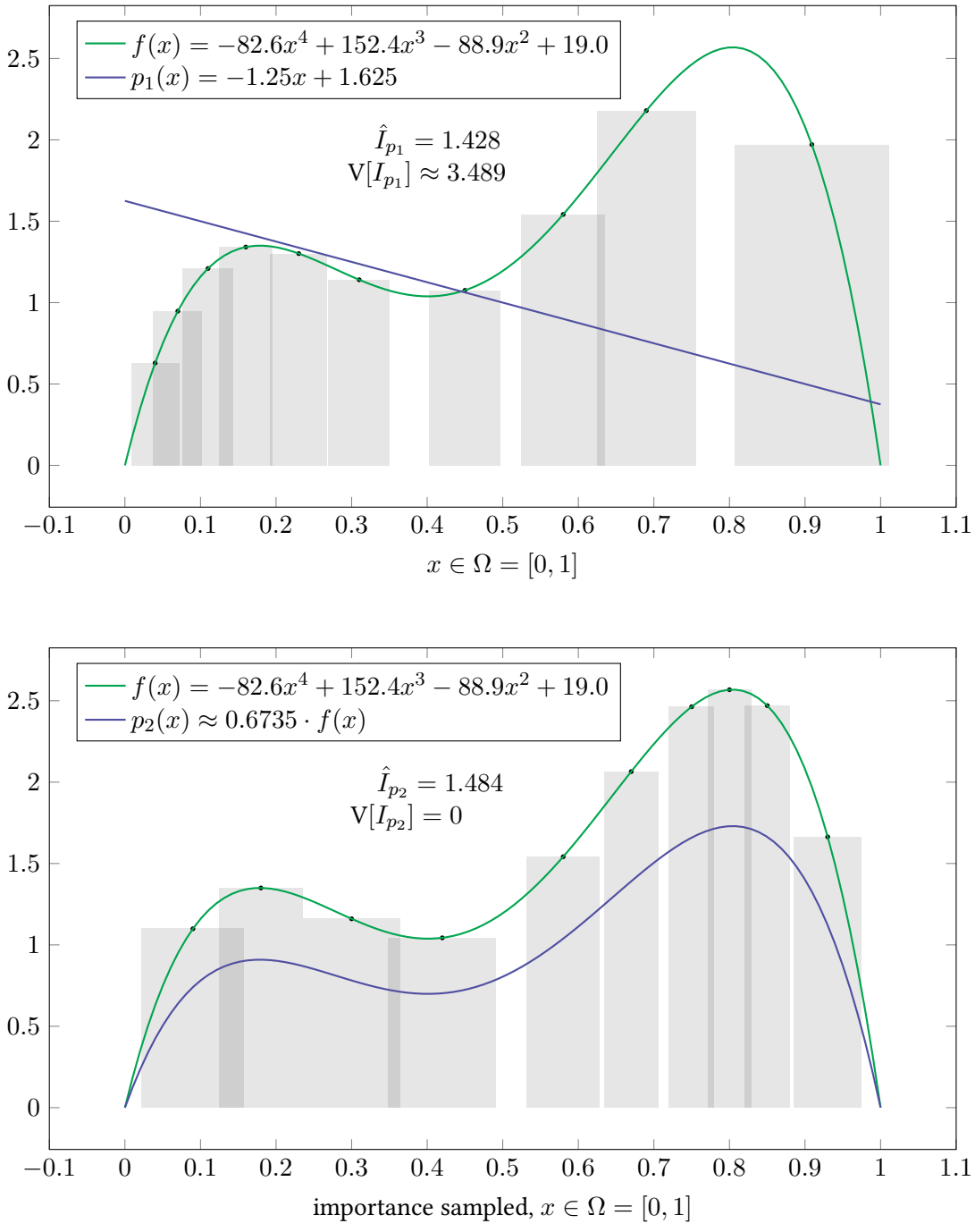


Figure II.4: Example of a Monte Carlo integration with 10 samples. The sum of the areas of the gray boxes is an estimate $\hat{I}$ of the integral of f. The width of each box is $1/Np(x)$. The correct value of the integral is ≈ 1.484 and the shown sampler variances are computed using Eq. (II.19).

In the case that $p(x) \propto f(x)$ the quotient of the two functions will become a constant c. It is easy to see that c can be factored out from both integrals (Eq. (II.17) and Eq. (II.19)) and that variance is zero in this case. That means, if we can sample with the PDF proportional to our searched function we get the final estimate after a single sample. In practice we do not know f in a closed form and are not able to sample with the ideal PDF. However, it is often possible to sample a part of the function ideally. For example, if the function is a product of two functions we might be able to sample one of them in an ideal fashion. Choosing a PDF similar to the target function is called *importance sampling* and is often worth its cost opposed to uniform sampling. An example for perfect sampling ($p \propto f$) is given in Figure II.4 at the bottom.

Equation (II.19) computes the variance of the estimator $\mathrm{V}[I]$ dependent on the PDF. By taking multiple samples N the variance is reduced over time. So, we are also interested in the sample variance $\mathrm{V}\left[\hat{I}_N\right]$ which is

$$\begin{aligned}
\mathrm{V}\left[\hat{I}_N\right] &= \mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right] \\
&= \frac{1}{N^2}\mathrm{V}\left[\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right] && \text{using Eq. (II.16b)} \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{V}\left[\frac{f(x_i)}{p(x_i)}\right] && \text{using Eq. (II.16a)} \\
&= \frac{1}{N}\mathrm{V}[I] && \text{(II.20)}
\end{aligned}$$

Since variance is a quadratic measure we need to take its square root, the standard deviation $\sigma = \sqrt{\mathrm{V}}$, to find the convergence rate of the expected sample value $\hat{I}_N$. Hence the convergence rate of a Monte Carlo sampler is $\mathcal{O}(N^{-0.5})$. That means the reduction of noise visible in a rendering is $1/\sqrt{N}$. For further details on numerical integration please refer to Veach's thesis [1997, Sec. 2.2].

II.2.3 Multiple Importance Sampling

Often, only parts of a function f are known or can be importance-sampled directly. In light transport, our target function is a product of the material response and the incoming light. For both there are importance sampling methods, of which none matches our entire target function. Instead, we can draw samples from two or more distributions and combine them in a weighted average. Thereby, the average is chosen to minimize the variance of the combined sampler. This process is called *Multiple Importance Sampling* (MIS).

Assume that $p_1, p_2, \ldots, p_S$ are the PDFs of different sampling methods from which $N_1, N_2, \ldots, N_S$ samples are drawn. Then there are many different ways to combine those samples into an unbiased estimator. For example each sampler can be used as stand-alone Monte Carlo estimator of

which the average over all S samplers

$$\hat{I}_{\text{MIS1}} = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{p_i(x_{ij})}$$

is again an unbiased estimate. Elvira et al. [2017] identified six basic strategies and analyzed their variance. If samplers are selected with replacement (i.e. for each sample we select one p_i randomly independent of the past) the best available strategy is to divide each sample by the sum of all possible PDFs:

$$\hat{I}_{\text{MIS2}} = \frac{1}{N}\sum_{i=1}^{S}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{\sum_{k=1}^{S} P_k \cdot p_k(x_{ij})}. \tag{II.21}$$

where $N = \sum_i^S N_i$ and $P_k = N_k/N$ is the probability to select the respective sampler. Indeed, this strategy is widely used in light transport simulation, albeit it is written in a slightly different form:

$$\begin{aligned}
(\text{II.21}) &= \frac{1}{N}\sum_{i=1}^{S}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{p_i(x_{ij})}\cdot\frac{p_i(x_{ij})}{\sum_{k=1}^{S} P_k \cdot p_k(x_{ij})}\\
&= \frac{1}{N}\sum_{i=1}^{S}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{p_i(x_{ij})}\cdot\frac{N_i \cdot p_i(x_{ij})}{N_i \cdot \sum_{k=1}^{S} \frac{N_k}{N} \cdot p_k(x_{ij})}\\
&= \frac{1}{N}\sum_{i=1}^{S}\frac{N}{N_i}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{p_i(x_{ij})}\cdot\frac{N_i \cdot p_i(x_{ij})}{\sum_{k=1}^{S} N_k \cdot p_k(x_{ij})}\\
&= \sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\frac{f(x_{ij})}{p_i(x_{ij})}\cdot w_i(x_{ij}) \qquad (\text{II.22})
\end{aligned}$$

where the last line is known as the *multi-sample* model. This form of multiple importance sampling on different transport methods was introduced by Veach and Guibas [1995a,b]. Veach also experimented with different weights w_i and found the same weight, namely

$$w_i(x) = \frac{N_i \cdot p_i(x)}{\sum_{k=1}^{S} N_k \cdot p_k(x)} \tag{II.23}$$

Balance Heuristic

which he called the *balance heuristic* as a result of an optimization process. The derivation can be found in Veach's thesis [1997, p. 288] and in the later Section II.7.6. The balance heuristic assigns large weights to samplers with a high probability density $p_i(x)$ compared to the other available samplers $p_k(x)$. Due to the higher likelihood, these samplers naturally have the smaller variance when producing sample x. Nevertheless, it is possible that none of the PDFs is a good match for the target function, resulting in a high variance regardless of the weight.

Weight optimization
Sec. II.7.6 p. 69

Since the balance heuristic is the result of an optimization process, it is the best choice to reduce variance in general, albeit it is not optimal. Veach provided the theoretical bounds and proved that no other heuristic can be significantly better than the balance heuristic (assuming positive weights). However, it is often said that the *power heuristic*

$$w_i^{\beta}(x) = \frac{(N_i \cdot p_i(x))^{\beta}}{\sum_{k=1}^{S}(N_k \cdot p_k(x))^{\beta}} \tag{II.24}$$

Power Heuristic

with the exponent $\beta = 2$ yields better results. It generalizes the balance heuristic and amplifies the decision between samplers of varying quality. This is most effective if one of the samplers is close to optimal, in which case the balance heuristic assigns too high weights to the other samplers.

Recently, Kondapaneni et al. [2019] observed that the optimal solution can be found if negative weights are allowed. Instead of being a weighted average, the solution becomes a linear combination of the sampling probability densities. While the optimal solution is a theoretical construct which requires integrated quantities, Kondapaneni et al. proposed an approximative method. It accumulates auxiliary values first, before the final integrated value can be directly computed as the solution of a system of equations.

Partial Sampling of the Domain

For a Monte Carlo estimator it is necessary that $p > 0$ everywhere in the domain where $f \neq 0$ to get a *consistent* sampler. With MIS Equation (II.21) it is possible to use samplers which sample only part of the domain. All we need is $f(x) \neq 0 \Rightarrow \sum P \cdot p(x) > 0$ which means that it suffices if one sampler is greater zero. This property can be extremely useful for the design of importance sampling methods, because we can construct complex functions piecewise over the domain.

Consistency II.2.2

II.2.4 Generating Random Numbers

To implement a Monte Carlo sampler we need tools to provide random numbers according to a given PDF on a computer. The drawn numbers must be statistically independent, which is necessary for an unbiased estimator. However, they do not need to be truly random. It is even possible to use pure deterministic sequences without breaking the estimator. Such an estimator is then called *Quasi-Monte Carlo* integrator.

Better and better pseudo *Random Number Generator*s (sRNGs) were developed in recent years. One important improvement was the *Permuted Congruential Generator* family from O'Neill [2014]. Its strength are a very small state, superior statistical properties and a very fast execution.

Another small state, high quality and fast generator is the *Xoroshiro128+* generator from Blackman and Vigna [2018]. In a small experiment both methods showed the same quality and performance. Hence, I decided to use the smaller one of the two: PCG 64 as shown in the listing. A small state is preferable if we have one generator per thread, which can be several thousands when working on a GPU. Further, it makes sense to use one generator per pixel to be able to rerun experiments deterministically by choosing the same seed (and being independent of the thread execution order). In this case the number of generators is as large as the number of pixels (often millions).

```
func pcg64(ref u64 state) -> u64
  x = state # Use previous state to reduce instruction dependency
  # Update the state (linear congruential generator)
  state = state * 6364136223846793005 + 1442695040888963407
  # RXS-M-XS output permutation
  x ^= x >> (5 + (x >> 59));
  x *= 12605985483714917081;
  return x ^ (x >> 43);
end
```

The pcg64 generator function provides 64 uniformly distributed bits with high quality. Since most sampling routines in rendering work on 32 bit floats only, one generator call is sufficient to generate two random numbers at the same time.

Additionally to the pseudo RNGs above, it is useful to generate low-discrepancy sequences. One method to reduce the variance of a sampler is to use stratified sampling. This can be achieved by algorithms which work on some kind of grid, or by using low-discrepancy sequences. Discrepancy is a measurement of the uniformity of a distribution. The smaller the discrepancy the more evenly spaced are the samples [Kuipers and Niederreiter 1974]. A good overview of the effectiveness of many different low-discrepancy series was presented by P. H. Christensen et al. [2018]. However, the sequence in the following was not analyzed in Christensen et al.'s survey.

The 1D sequence with a very small discrepancy is the additive recurrence series

$$s_{n+1} = (s_n + \Phi) \bmod 1 \tag{II.25}$$

$x \in \mathbb{R} \bmod 1$ is the fractional part

where $\Phi = (1 + \sqrt{5})/2$ is the golden ratio. Unfortunately, we need to generate higher dimensional samples for most of our applications. One possible generalization of the additive recurrence series is to use different irrational numbers (e.g. the square roots of primes) to replace Φ. Also, a generalization of the golden ratio sequence to 2D was made by Schretter et al. [2012] by using permutations. Another option with a higher quality is to distribute the golden ratio sequence along a space filling curve [Schretter et al. 2016]. This idea can be extended to higher dimensions, but this was not evaluated by the authors.

In this thesis, I use the 2D golden ratio sequence along the Hilbert curve ([Schretter et al. 2016]) to generate directions from the camera. It is reasonable to use the high quality stratified sequence for the camera, because the gain at the beginning of a path is the highest. If used later on a path, the randomization from previous steps reduces the gain of high quality random numbers. In the case of later events the PCG 64 provides very good statistical properties and stratified samples would give a smaller advantage.

II.2.5 Inverse CDF Method

Now that we have a set of pseudo random number sources we get standard uniformly distributed variables $\mathcal{U} \in [0, 1)$. However, as shown in Figure II.4, we would like to sample according to some other distribution function. A generic method to transform uniform samples into a different distribution is the inverse *Cumulative Distribution Function* (CDF) method.

Let $p : [a, b] \mapsto [0, \infty]$ be our target distribution density function with a CDF

$$F(x) = \int_a^x p(t)\,\mathrm{d}t, \tag{II.26}$$

then $F^{-1}(\mathcal{U})$ produces samples according to p. Figure II.5 visualizes this process. If p is large, F increases faster. This leads to a compression of the interval on the $\mathcal{U}$ axis onto the smaller x-axis, i.e. an increase of the sample density.

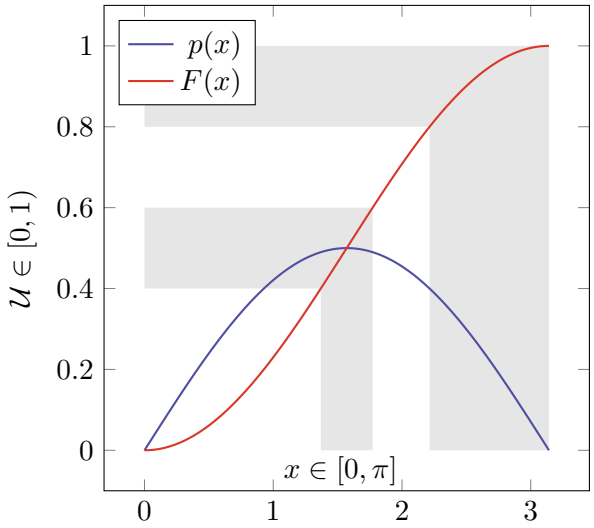


Figure II.5: Example of inverse CDF method for $p(x) = \sin(x)/2$. The two intervals on the $\mathcal{U}$-axis are equally wide. Their size on the x-axis depends on the slope of the CDF which is $p(x)$ integrated over the interval.

The example function in the figure is the sinus which gives:

$$\begin{aligned} p(x) &= \frac{\sin x}{2} \\ F(x) &= \frac{1 - \cos x}{2} = \mathcal{U} \\ F^{-1}(\mathcal{U}) &= \arccos(1 - 2\mathcal{U}) \end{aligned}$$

for the desired transformation. In the example, integration and inversion are rather simple. For some PDF, however, any of the two steps may be impossible to compute in a closed form, for example the integration of the Gaussian distribution. In such cases simplification and non-generic approaches might still lead to feasible algorithms. Examples are the Box-Muller transform [1958] to sample Gaussian distributions or the sampling of visible normals which will be shown in Section II.5.5.

Inverse CDF in Two Dimensions

Most of our target PDFs are two-dimensional, since we are interested in sampling directions (in θ, ϕ). We cannot directly solve the CDF for both dimensions simultaneously, but we can do so sequentially. Let $p : [a_x, b_x] \times [a_y, b_y] \mapsto [0, \infty]$ be our two-dimensional PDF. The CDF is obtained by integrating over both variables:

$$F(x, y) = \int_{a_y}^{y} \int_{a_x}^{x} p(s, t) \, \mathrm{d}s \, \mathrm{d}t. \tag{II.27}$$

The process described in the following is visualized in Figure II.6. First, it is possible to fix one of the variables to its maximum value. W.l.o.g. I set y (opposed to x) to its maximum value b_y. This defines a new 1D CDF $F(x, b_y)$ over the variable x. $F(x, b_y)$ is the probability to be anywhere in the rectangle left of x as depicted in Figure II.6 left. Thus, the inserve CDF

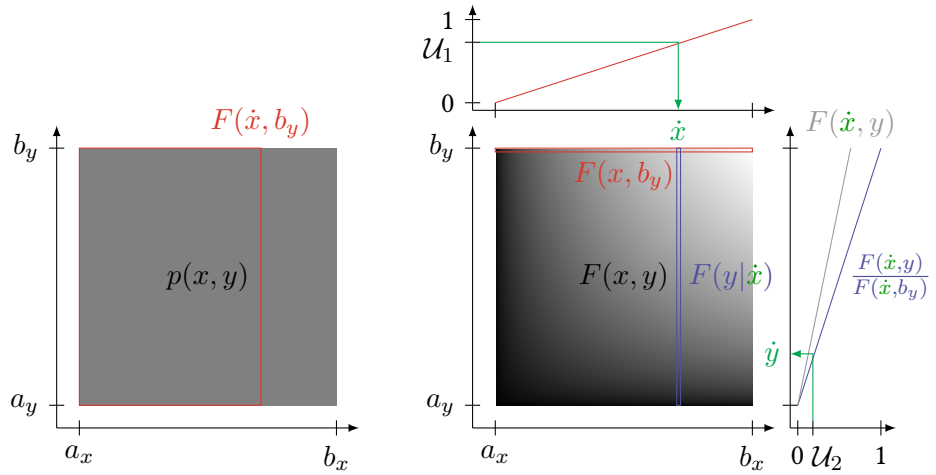


Figure II.6: 2D inverse CDF example of the uniform density $p(x, y)$ (left). The corresponding CDF (right) grows linearly in x and y. The red plot shows the top line $F(x, b_y) \in [0, 1]$ of the 2D function. Here, the 1D inverse CDF can be applied and yields x'. The function $F(x', y)$ is not a CDF since it does not reach 1 as maximum value. However, the conditional $F(y|x')$ normalizes the selected vertical line and the 1D inverse CDF can be applied again to get y'.

can be applied as before

$$\begin{aligned} F(x, b_y) &= \mathcal{U}_1 \\ \dot{x} &= F^{-1}(\mathcal{U}_1, b_y) \end{aligned} \tag{II.28}$$

to find the first variable. For the second variable we can insert the new $\dot{x}$ and need to sample the conditional probability density

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(x, y)}{\int_{a_y}^{b_y} p(x, u)\,\mathrm{d}u}. \tag{II.29}$$

Applying some transformations on the conditional cumulative density function $F(y|x)$, it can be expressed with respect to the known $F(x, y)$:

$$\begin{aligned} F(y|x) &= \int_{a_y}^{y} \int_{a_x}^{x} p(t|s)\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_{a_x}^{x} \int_{a_y}^{y} \frac{p(s, t)}{p(s)}\,\mathrm{d}t\,\mathrm{d}s \\ &= \int_{a_x}^{x} \frac{1}{p(s)} \int_{a_y}^{y} p(s, t)\,\mathrm{d}t\,\mathrm{d}s \\ &= \frac{\int_{a_x}^{x} \int_{a_y}^{y} p(s, t)\,\mathrm{d}t\,\mathrm{d}s}{\int_{a_x}^{x} \int_{a_y}^{b_y} p(s, u)\,\mathrm{d}u\,\mathrm{d}s} \\ &= \frac{F(x, y)}{F(x, b_y)}. \end{aligned} \tag{II.30}$$

Finally, $\dot{y}$ can be found by computing $F^{-1}(\dot{x}, \mathcal{U}_2 \cdot F(\dot{x}, b_y))$. It is possible to proceed in the same manner for even more dimensions, but we will only need the inverse CDF method for up to two dimensions.

II.2.6 PDF UNDER CHANGE OF VARIABLES

Many of the more complex sampling algorithms transform a uniform variable into some desired output distribution. This includes, for example, the inverse CDF method and the reflection/refraction at half vectors as will be described in the materials section. Often, it is also important to determine the PDF of a given algorithm. The tool to achieve that is to use the determinant of the Jacobian of the algorithm.

Half vector sampling II.5.5 p. 51

Let $Y = g(X)$ be the bijective, differentiable operator performed by the algorithm. Then the probability inside a differential area must be invariant under this transformation:

$$\begin{aligned} & |p_Y(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}| = |p_X(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}| \\ \Leftrightarrow \quad & p_Y(\boldsymbol{y}) = \left|\det \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{y}}\right| p_X(\boldsymbol{x}) \end{aligned}$$

Note that both X and Y can be multivariate distributions. To be able to compute the derivatives and the final PDF value we need the bijective property $\boldsymbol{x} = g^{-1}(\boldsymbol{y})$ which leads to

$$\begin{aligned} p_Y(\boldsymbol{y}) &= \left|\det \frac{\mathrm{d}g^{-1}(\boldsymbol{y})}{\mathrm{d}\boldsymbol{y}}\right| p_X(g^{-1}(\boldsymbol{y})) \\ &= \left|\det \boldsymbol{J}_{g^{-1}}(\boldsymbol{y})\right| p_X(g^{-1}(\boldsymbol{y})) \end{aligned} \tag{II.31}$$

Usually, we already know $p_X(g^{-1}(\boldsymbol{y}))$ which is the PDF value of our sample before applying the transformation g. What remains is to evaluate the Jacobian $\boldsymbol{J}_{g^{-1}}$ for the created sample. The Jacobians used in this thesis are derived in appendix A.1 p. 191.

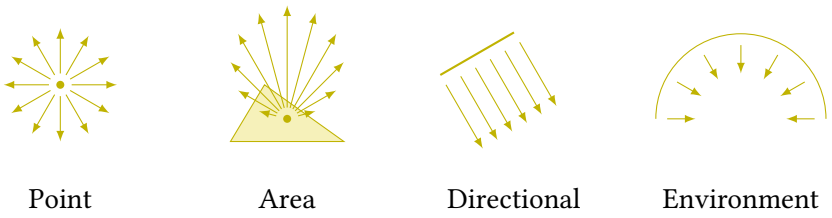


Figure II.7: Types of light sources.

II.3 Scene Modeling: Light Sources

A scene consists of a geometrical description for the surfaces, materials to describe the interaction of light with the matter and the lights themselves. Last of all, a camera model is required to capture the image.

This and the following sections introduce the models used in this thesis. It is not a complete introduction to possible representations. Further alternative models are only partially mentioned.

Section II.1 introduced the formulas to describe light radiation in general. A light source, as described by this section, has special restrictions and defines the use of the above equations. To be able to use a light source in all kinds of light transport algorithm, it needs to define three operations:

Radiometric Quantities Sec. II.1 p. 13

Requirements of light sources

Direct illumination must return the differential irradiance (Equation (II.5)) for some given point x. Further, it needs to evaluate the sampling probability density p as if the outgoing direction would be sampled.

Sampling must randomly create a new photon according to the characteristic density. It needs to return the flux ϕ, the ray $(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow)$ and the sampling probability density p.

On hit the radiance L and the probability density p are required. This operation occurs if a random walk ends on a light source by chance.

Random walk Sec. II.7.3 p. 65

The probabilities are necessary for the MIS computations which were introduced in Section II.2.3. For methods which do not use MIS this quantity is not required.

In the description of the direct illumination $\mathrm{d}E(\boldsymbol{x}, \boldsymbol{x}_L)$ will be used instead of the form $\mathrm{d}E(\boldsymbol{x}, \boldsymbol{d}_\downarrow)$, because the receiver location $\boldsymbol{x}$ is a point on the ray $(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow)$. I.e. the incident and the excitant directions are defined by $\boldsymbol{x} - \boldsymbol{x}_L$.

II.3.1 Point Lights

A point light is a single point emitting flux uniformly to all directions. While this is one of the conceptually simplest light sources, it is not physically plausible due to its infinitesimal extent. In consequence, it has an infinite *radiance* and cannot be hit randomly. For rendering this means that a special treatment becomes necessary in some situations.

The point light is parametrized by its position $\boldsymbol{x}_L$ and its flux Φ, leading to the following radiometric formulas

$$L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) = \infty$$
$$\mathrm{d}E(\boldsymbol{x}, \boldsymbol{x}_L) = \frac{\Phi \left|\cos\theta_\downarrow\right|}{4\pi \left\|\boldsymbol{x} - \boldsymbol{x}_L\right\|^2}$$
$$p(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) = \frac{1}{4\pi\,\mathrm{sr}}$$

$\cos\theta_\downarrow$ is the incident cosine at $\boldsymbol{x}$
$\|\cdot\|$ is the Euclidean norm

The $4\pi\,\mathrm{sr}$ is the maximum solid angle, resulting from the uniform distribution over the sphere. To sample such a direction, the following algorithm, obtained by the inverse CDF method, can be used

Inverse CDF II.2.5 p. 25

Sampling of a uniform direction on the unit sphere

```
func sampleUniformDir(𝒰₁, 𝒰₂) -> d
  cosTheta = 𝒰₁ * 2 - 1
  sinTheta = sqrt(1 - cosTheta^2)
  phi = 𝒰₂ * 2 * π
  return [sinTheta * sin(phi), sinTheta * cos(phi), cosTheta]
end
```

II.3.2 Area Lights

A more plausible type of light source is the area light. Typically, it is modeled as a surface with an angular constant *radiance* L, also known as a Lambert emitter. That means it exhibits an isotropic behavior and appears equally bright as seen from each direction. For practical reasons it is useful to restrict the emittance to the upper half-space of the surface only. The lower hemisphere, opposite to the normal, is often inside of a closed model and would not contribute to the image seen outside.

In this thesis, the shape of an area light is restricted to that of a triangle. Therefore, it is parametrized by its *radiance* L and three vertex positions for the triangle $\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C$. From the three positions, the triangle normal and the area A can be computed using the cross product. Due to its extent, it is necessary to sample the position $\boldsymbol{x}_L$ on the triangle for both direct illumination and sampling. Thus, the resulting formulas for direct illumination and sampling are:

$$L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) = \text{constant} = \frac{I(\boldsymbol{d}_\uparrow)}{A \cdot \cos\theta_\uparrow} \tag{II.32a}$$

$L = I/(A\cos\theta)$ from Eq. (II.8) p. 17

$$\mathrm{d}E(\boldsymbol{x}, \boldsymbol{x}_L) = \frac{L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) \cdot A \cdot \cos\theta_\uparrow \cdot \left|\cos\theta_\downarrow\right|}{\left\|\boldsymbol{x} - \boldsymbol{x}_L\right\|^2} \tag{II.32b}$$

Note that $I(\boldsymbol{d}_\uparrow) = I_0 \cos\theta_\uparrow$ due to Lambertian property; thus the varying $\cos\theta_\uparrow$ in (II.32a) cancels out

$$p(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) = p(\boldsymbol{x}_L) \cdot p(\boldsymbol{d}_\uparrow) = \frac{1}{A} \cdot \frac{\cos\theta_\uparrow}{\pi\,\mathrm{sr}} \tag{II.32c}$$

An important thing to notice is that there are two sampling events. The first uniformly samples a position on the triangle and is even required for direct illumination. The second sampling procedure produces the outgoing direction according to a cosine distribution, because the intensity of a Lambert emitter follows this distribution.

To uniformly sample the position on a triangle there are two possible algorithms. Both produce a barycentric coordinate (α, β, γ) which can be used to interpolate the vertices (positions and further parameters) of the

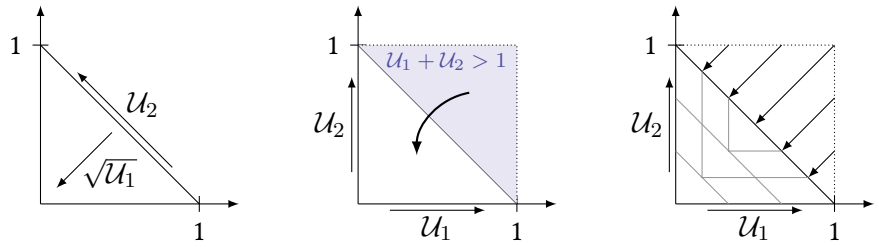


Figure II.8: Alternative approaches of sampling a triangle uniformly. Left: the well known closed form solution. Center: alternative using a reflection at the diagonal. Right: alternative using a distortion mapping quad → triangle.

triangle. Barycentric coordinates itself are local triangle coordinates which form a linear combination of the associated vertices, as for example $\boldsymbol{x}_L = \alpha\boldsymbol{x}_A + \beta\boldsymbol{x}_B + \gamma\boldsymbol{x}_C$.

Both of the following algorithms were introduced in the article *Generating Random Points in Triangles* [Turk 1990]. The most known algorithm computes the barycentric coordinate in a closed form solution.

Closed form sampling of a barycentric coordinate

```
func sampleTriangle1(𝒰₁, 𝒰₂) -> [α,β,γ]
  t = sqrt(𝒰₁)
  return [1 - t, t * (1 - 𝒰₂), t * 𝒰₂]
end
```

A different approach can be taken by sampling the unit square and reflecting the points in the wrong half. The two ideas are compared in Figure II.8 visually. The advantage of the second idea is that it does not need to compute a square root and could be faster, depending on the architecture.

Sampling of a barycentric coordinate by reflection

```
func sampleTriangle2(𝒰₁, 𝒰₂) -> [α,β,γ]
  if 𝒰₁ + 𝒰₂ > 1: # mirror coordinate
    𝒰₁ = 1 - 𝒰₁
    𝒰₂ = 1 - 𝒰₂
  end
  return [1 - (𝒰₁ + 𝒰₂), 𝒰₁, 𝒰₂]
end
```

Note that the branch usually becomes a conditional move and is therefore rather efficient. However, a small benchmark on a current Intel CPU has shown the same performance for both methods. Since performance also depends on instruction dependencies and on utilization of certain *Arithmetic Logical Unit* (ALU) instructions (like the square root) this may be different in another context or architecture. If the sampling of triangles becomes a bottleneck in some application, the second method should be tried as an alternative, otherwise sticking to the previous algorithm is just fine.

Recently, Heitz [2019] introduced a third alternative.

Sampling of a barycentric coordinate by distorting

```
func sampleTriangle3(𝒰₁, 𝒰₂) -> [α,β,γ]
  if 𝒰₂ > 𝒰₁:
    𝒰₁ *= 0.5
    𝒰₂ -= 𝒰₁
  else:
    𝒰₂ *= 0.5
    𝒰₁ -= 𝒰₂
  end
  return [1 - (𝒰₁ + 𝒰₂), 𝒰₁, 𝒰₂]
end
```

He argues that the square root approach destroys the distribution properties (for example blue noise or stratification patterns). His alternative approach remaps a quad to the triangle by moving one vertex inwards, producing a higher quality mapping. However, he did not compare the new approach to `sampleTriangle2` which has similar properties.

Finally, to create a photon from an area light source, we also need a sampler for a cosine distributed direction with density $p_{\cos}(\boldsymbol{d}_\uparrow) = \cos\theta_\uparrow/\pi\,\mathrm{sr}$. This can be found using the inverse CDF method again.

Def. $p_{\cos}(\boldsymbol{d}_\uparrow)$

Inverse CDF II.2.5 p. 25

Sampling of a cosine distributed direction

```
func sampleCosineDir(ξ1, ξ2) -> d
  cosTheta = sqrt(ξ1)      # cos(acos(sqrt(x)))
  sinTheta = sqrt(1 - ξ1) # sin(x) = sqrt(1-cos(x)^2)
  phi = ξ2 * 2 * π
  return [sinTheta * sin(phi), sinTheta * cos(phi), cosTheta]
end
```

II.3.3 Directional Lights

Directional lights are again artificial constructs, like the point lights before. They are often used to model distant light sources, like the sun, for which parallel rays can be assumed. Additionally, they are an essential element of the environment light sources which also assume an infinite distance between receiver and light source. Because of that assumption, they are also called *distant lights* in the PBRT book [Pharr et al. 2017].

Assumption: Infinite distance ⇔ parallel light.

To model a directional light we use a direction $\boldsymbol{d}_L = \boldsymbol{d}_\uparrow$ and define the *radiance* L directly.

$$L(\boldsymbol{x}_L, \boldsymbol{d}_L) = \text{constant}$$
$$\mathrm{d}E(\boldsymbol{x}, \boldsymbol{d}_\downarrow) = L(\boldsymbol{x}_L, \boldsymbol{d}_L) \cdot |\cos\theta_\downarrow| \cdot 1\,\mathrm{sr}$$

Note that $\mathrm{d}E$ misses the *solid angle* $\mathrm{d}\omega$ from the previously used Equation (II.9). This description is possible because for parallel rays the density of light does not change over the distance. In a mathematical way it also makes sense, since in the integral only a single direction is not zero:

$$E(\boldsymbol{x}) = \int_\Omega L(\boldsymbol{x}, \boldsymbol{d}_\downarrow) \cdot \delta(\langle \boldsymbol{d}_L, \boldsymbol{d}_\downarrow \rangle) \cdot |\cos\theta_\downarrow| \,\mathrm{d}\omega$$

where $\delta(x)$ is the *Dirac delta function* which is zero everywhere except for $x = 0$ where it becomes infinite. The integral $\int_\Omega \delta(\langle \boldsymbol{d}_L, \boldsymbol{d}_\downarrow \rangle)\,\mathrm{d}\omega$ is equal to one steradian and only the value $L(\boldsymbol{x}, \boldsymbol{d}_\downarrow = \boldsymbol{d}_L)$ is multiplied with non-zero in the integral. Therefore, the above equation for $\mathrm{d}E$ holds.

Introduction of Dirac delta function

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$$

To define a sampling probability we first need an idea on how to sample a directional light source. Choosing the direction is a deterministic event without any requirement for sampling. On the other hand, the ray origin must allow to hit any point in the scene, which is visible from the given direction. Hence, one option would be to sample an arbitrary surface point and to offset this in negative ray direction until it is outside the scene. However, sampling a point on the surface is too expensive and would require to find the PDF which depends globally on the scene. Luckily, there is a simpler

Sampling model for directional lights

possibility: Using the scene's *bounding box* (BB) directly allows to sample points outside the scene. For any given direction, at most three of the box-sides are visible and it is simple to distribute photons uniformly over these three surfaces. Hence, we have a random sampling over the projected area $A_{\mathrm{BB}_\perp}$ of the bounding box

$$p(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) = \frac{1}{A_{\mathrm{BB}_\perp}}$$

Some of the resulting rays may miss the scene, but the result will be correct as long as any point, visible from the light source, can be hit. The important thing is that the seeding area is as least as large as the projected scene. To improve efficiency by avoiding misses, the area should be as small as possible. A different avenue would be to use a disc derived from the *bounding sphere*. However, in most cases the *bounding box* fits the scene more closely than the *bounding sphere*.

The algorithm to generate a position on a projected *bounding box* looks as follows

Sampling of a position on a projected BB

```
func samplePosOnBoundary(d, BB, U1, U2) -> x_L, p
  # Compute partial sums of projected areas.
  Ax   = BB.sideLen.y * BB.sideLen.z * abs(d.x)
  Axy  = BB.sideLen.x * BB.sideLen.z * abs(d.y) + Ax
  Axyz = BB.sideLen.x * BB.sideLen.y * abs(d.z) + Axy
  # Get a position in [0,1]^3. To choose a face treat
  # all areas as connected in dimension U1.
  if   U1 < Ax / Axyz:  p = [step(-d.x), rescale(U1,0,Ax), U2]
  elif U1 < Axy / Axyz: p = [rescale(U1,Ax,Axy), step(-d.y), U2]
  else:                 p = [rescale(U1,Axy,Axyz), U2, step(-d.z)]
  return (BB.min + p * BB.sideLen, 1 / Axyz)
end
```

step(x) = if x > 0: 1 else: 0
rescale(x,a,b) = (x-a) / (b-a)

The helper function `step` choses between two opposite sides depending on the direction. `rescale` remaps a value from a range $[a, b]$ to $[0, 1]$. This trick is used to make use of a single random variable for the decision process and the sampling inside the range. Otherwise three random numbers would have been necessary. By rescaling the random number it loses part of its precision. However, reusing the number only once with just a few similar sized intervals does not cause problems in practice.

II.3.4 Environment Lights

Environment lights can greatly enhance the visual appearance by adding a natural component of background illumination. The idea is to map an image to an infinite distant hull. It was first introduced to computer graphics by Blinn and Newell [1976] to render more realistic specular reflections.

The environment map is seen equally from all points and is not subject to perspective effects. Thus, each single direction in the environment map can be treated like a directional light from the previous section. A difference is that this time it is possible to hit the environment map randomly. Further, the sampling process now needs to sample the direction, too.

$$\begin{aligned} L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) &= \text{texture} \\ \mathrm{d}E(\boldsymbol{x}, \boldsymbol{x}_L) &= L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) \cdot |\cos\theta_\downarrow| \cdot \mathrm{d}\omega \\ p(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow) &= p(\boldsymbol{d}_\uparrow) \cdot p(\boldsymbol{x}_L) = p(\boldsymbol{d}_\uparrow) \cdot \frac{1}{A_{\mathrm{BB}_\perp}} \end{aligned}$$

Similar to the directional light, $\mathrm{d}E$ depends on the *radiance* without notion of a distance. Although the radiance is different for each $\boldsymbol{d}_\uparrow$, light from each single direction is still treated as a parallel source. Moreover, it is necessary to sample $p(\boldsymbol{d}_\uparrow)$ in the context of direct illumination, now.

The probability density $p(\boldsymbol{d}_\uparrow)$ for the excitant direction is defined by the texture. It is the value of the texture divided by the integral $\int_\Omega L(\boldsymbol{x}_L, \boldsymbol{d}_\uparrow)\,\mathrm{d}\omega$. I.e. the texture must be normalized to integrate to one to be used as a probability density.

To sample this texture the inverse CDF method can be applied again. In this case the CDF of the discrete texture is an array of the same size containing the prefix sums up to each pixel. Instead of an explicit inversion of the CDF, a binary search must be executed for each new sample to find the position on the texture. A detailed description of such a sampling algorithm can be found in [Pharr et al. 2017, pp. 245–250].

Further Reading

Spectral effects of light sources

All introduced light sources are fairly simple models which suffice to stress test all kinds of transport algorithm. In reality, a light source often has a characteristic over different wavelengths λ and is rarely as uniform over area or direction as the models in this section. Especially, the physical reasons for radiation are omitted here.

Radiant exitance M Eq. (II.7) p. 16

A fundamental effect is that of blackbody radiation. Any body with a temperature above absolute zero (0 K) radiates energy as expressed by the *Stefan-Boltzmann law* $M = \sigma T^4$ where $\sigma \approx 5.67 \times 10^{-8}\mathrm{Wm}^{-2}\mathrm{K}^{-4}$ is the *Stefan-Boltzmann constant* and T the temperature in Kelvin. The spectral distribution of that radiation is itself described by *Planck's law* and also changes over temperature. Beyond that, the optical and chemical properties influence the emitted spectral distribution. Details on these topics can be found in [McCluney 1994] Chapter 3 and 6.

Practical spectral models

Another more practical solution to describe the spectrum of a real emitter was introduced by the CIE in 1931 [Guild et al. 1931]. They defined the *Standard Illuminants* for different light situation. For example the *D65 illuminant* describes mid-day sunlight in Europe. A short introduction to that topic can be found in [Pharr et al. 2017] Chapter 12 as well as in [McCluney 1994, pp. 369–370].

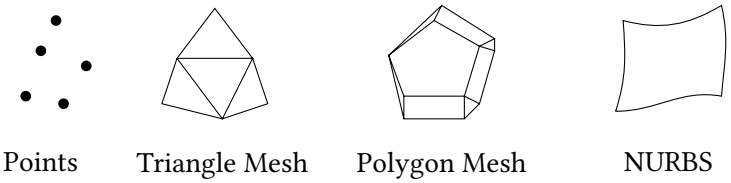


Figure II.9: Alternative primitives to define geometry.

II.4 Scene Modeling: Geometry

In computer graphics, geometry is often modeled by small discrete primitives, from which the triangle is the most commonly used one. There are further options like spheres, quads and more advanced patches like NURBS (Non-uniform rational B-Splines). Also, point clouds are the natural result of 3D scanning processes and are used for sparse sampling of the scene surface in the computation of indirect light. However, the triangle is the simplest primitive which can be used to model closed surfaces. It also has a fast closed-form intersection test (e.g. [Möller and Trumbore 1997]) which is required to find intersections between particle paths (rays) and the geometry.

Generally, the primitives are connected in a graph structure called *mesh*. It is possible that information is shared between primitives over corners (vertices) or edges. For rendering, data is usually stored per triangle and per vertex. Storing information on edges is less common in real-time graphics, but is used as well for CAD applications.

Data storage on triangle meshes

On the triangle basis there is information like the associated material and the geometrical tangent frame. As part of the tangent frame, the *geometric normal* (triangle normal) determines inside and outside as defined by the winding order. Typically, the normal points towards the observer (front face) if the vertices are seen counterclockwise.

Data stored at vertices is often shared between triangles and interpolated over the triangle using barycentric coordinates. Beside the mandatory positions this often includes the *shading normals* and the texture coordinates.

Another important thing to model is the interior of objects. Light is scattered and absorbed inside a medium which must be defined. While this thesis mainly focuses on surface transport, where light only interacts with the surfaces, it should be noted that geometry must be watertight to support this kind of information. In a watertight mesh no point of the interior can be reached from outside without having a surface intersection. Otherwise, there would be rays which cannot be assigned to a volumetric material uniquely. Inhomogeneous media can be modeled by volume textures without restricting geometry. Often a combination of both is used. For example to define the medium of glass objects, it is more precise to assign this information to the (interior) material of the mesh. For inhomogeneous phenomena like fog a volume texture is preferred.

Note on volumetric properties

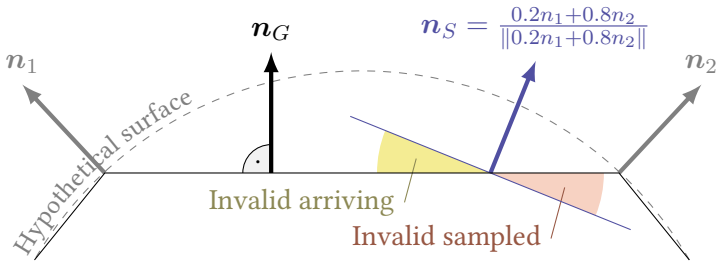


Figure II.10: Shading normals n_S are used to emulate smooth surfaces.

II.4.1 Shading Normals

Due to the discretization to linear triangles, smooth surfaces are a problem. They either look faceted or require an incredibly high amount of primitives. While NURBS model those surfaces better, they lack simple intersection methods. A solution is to store normals at the vertices and use interpolation to generate the *shading normals*. These new normals continue smoothly over edges.

By pretending that the interpolated *shading normal* is the normal n in the formulas of the previous and the upcoming sections, we obtain the appearance of a smooth surface. To differentiate the normals where necessary, n_G is used for the *geometric normal* and n_S for the *shading normal*. Figure II.10 shows the relation of these vectors.

Asymmetry of shading normals (real ↔ expected photon density)

Unfortunately, this trick also causes an asymmetry in the light transport. The density of received particles depends on the cosine of the real surface, as described in Section II.1.3. Thus, the density of traced particles depends on n_G. On the other hand, when computing the *irradiance* directly with n_S, a different particle density is assumed.

Using shading normals in a path tracer (i.e. only view path tracing and direct illumination) results in the desired smooth appearance. Therefore, we want to modify *photon* paths to match the behavior of shading normals on the view sub-path.

A solution was given by Veach [1997, pp. 150–158] who enforced the reciprocity of light transport. First, the deviation in the density can be corrected by multiplying the transported *flux* with

$$\max\left(0, \frac{\langle n_S, d_\downarrow\rangle}{\langle n_G, d_\downarrow\rangle}\right).$$

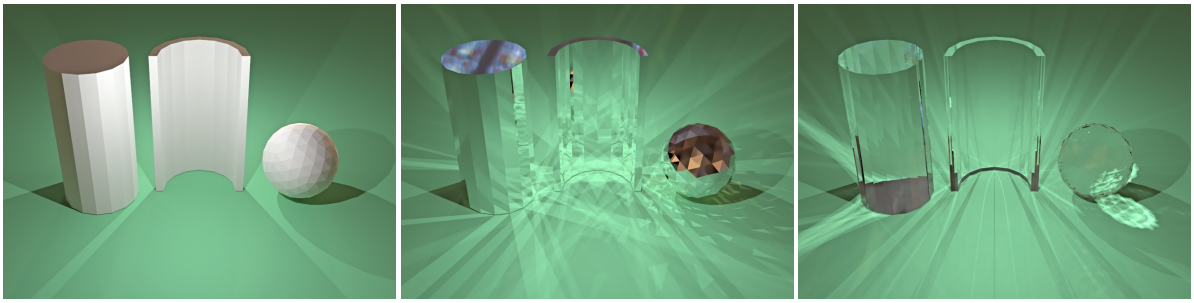
I.e. by dividing with the real density scale $\langle n_G, d_\downarrow\rangle$ and multiplying with the new scale $\langle n_S, d_\downarrow\rangle$, the *radiant power* is scaled to fit the expectation of the shading normal.

The clamping at zero is necessary because photons can arrive from below the virtual surface while being above the real surface, as shown by the yellow area in Figure II.10. These paths are invalid and must be discarded. In the same way, continuing the path by random sampling might create invalid paths too (red area in Figure II.10).

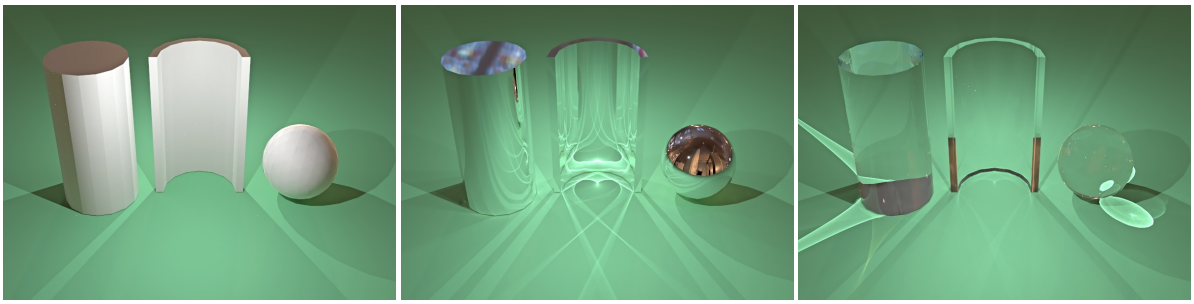
However, using the above term for the *flux* only, would still violate the symmetry, because the density of outgoing directions is not corrected. Thus, the full correction term reads

$$\max\left(0, \frac{\langle n_S, d_\downarrow\rangle}{\langle n_G, d_\downarrow\rangle}\right) \cdot \max\left(0, \frac{\langle n_G, d_\uparrow\rangle}{\langle n_S, d_\uparrow\rangle}\right) \tag{II.33}$$

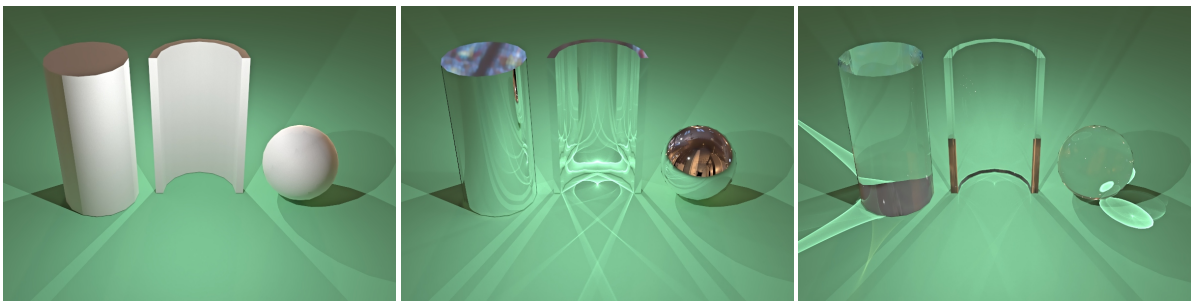
Def. Shading normal correction term



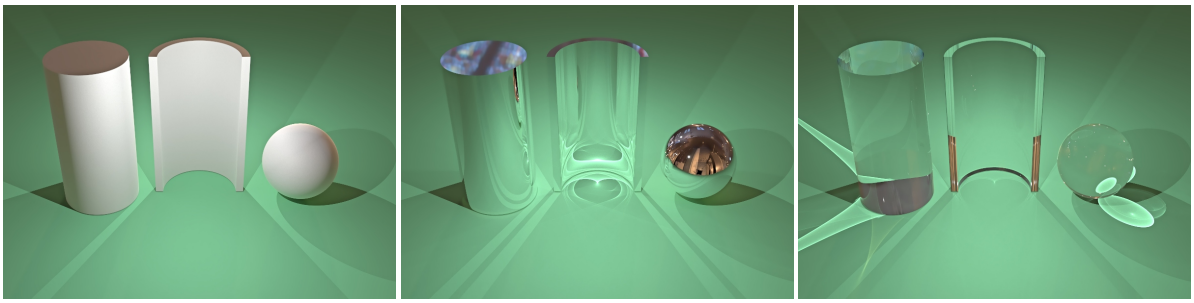
Flat: using geometric normal $\boldsymbol{n}_G$ (454 triangles)



Shading normals $\boldsymbol{n}_S$ without correction (454 triangles)



Shading normals $\boldsymbol{n}_S$ with correction (454 triangles)



Ground truth: highly tessellated polygons with corrected $\boldsymbol{n}_S$ (83974 triangles)

Figure II.11: Effects of shading normals rendered with 5000spp VCM* (Section IV.3 p. 107). The three objects are illuminated by two point light sources and an environment map. Without correction (second row) the symmetry breaks and facets become slightly visible on diffuse surfaces. The correction (Equation (II.33), third row) fixes the problems of asymmetry, but cannot remove artifacts in caustics. Moreover, it even introduces new artifacts at the edges of specular objects (see metal and glass cylinder).

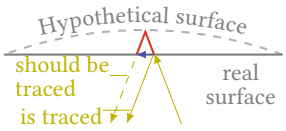
and must be multiplied on each sampling and evaluation event on a light sub-path only.

Note that for merges, which will be introduced in Section II.7.5, two different *geometric normals* must be used. In a merge, we pretend that two sub-path end points are at the same location, though they both could end on different triangles with different normals $\boldsymbol{n}_G$. In this case, $\langle \boldsymbol{n}_G, \boldsymbol{d}_\downarrow \rangle$ must use the normal at the light sub-path, whereas $\langle \boldsymbol{n}_G, \boldsymbol{d}_\uparrow \rangle$ must use the normal at the view sub-path. The shading normal is still the same for both terms, namely the one at which the shading is preformed, which is usually the end of the view sub-path.

Merges II.7.5 p. 68

Figure II.11 shows the utility of *shading normals* in bidirectional tracing algorithm. Using corrected *shading normals* increases the virtual surface smoothness on rough surfaces. Without correction the result on diffuse surfaces still shows faceting from light tracing events. The part of the result generated by path tracing already appears smooth without the correction. On the other hand, the correction introduces dark halo artifacts where clamping applies. I.e. in cases where the two normals are oriented differently with respect to the ray. This is visible on the boundaries of the specular objects.

Unfortunately, the correction term is not able to handle all errors in reflected photon densities, as visible in the caustics. It only corrects the error made by the projective area, leading to minor improvements (e.g. in front of the metal cylinder). It does not handle the change in path lengths (side image, red) or path offset (blue arrow), which cause a significant difference in the caustics compared with the ground truth.



The ground truth in Figure II.11 (last row) was rendered with $185\times$ as many primitives which needed $\approx 15\%$ more time. However, using more triangles will have diminishing returns for most situations. To emphasize the artifacts, the first three rows were rendered with an extremely rough tessellation. In practice a medium tessellation (around ≈ 2000 triangles) would lead to acceptable results.

There are other solutions to the shading normal problem. For example, in *Consistent Normal Interpolation* [Reshetov et al. 2010] normals are modified to always show into the same direction as the geometric normal with respect to the current directions. With their approach the clamping ($\max(0, \cdot)$) is not necessary, while the other terms are still needed to make the light transport symmetric.

Alternative solutions for correct shading normals

A more fundamental solution is to modify the material model as in *Microfacet-based Normal Mapping* [Schüssler et al. 2017]. The trick is to modify the microfacet normals (see next section) such that the average normal points into the desired direction. By simulating infinite bounces between the microfacets, it is possible to solve the asymmetry in an energy conserving way. However, according to Fig. 18 in Schüssler et al.'s paper this is still not sufficient to solve all invalid cases in the shading normal application.

II.5 SCENE MODELING: MATERIALS

Now that there are light sources and a description of surfaces, we need a model of how light interacts with matter. A material specifies the optical properties of the volume and its interface. However, a common practice in graphics is to split the description of a material into several parts – its surface and the medium inside the volume.

For surfaces the most used form is the *Bidirectional Reflectance Distribution Function* (BRDF, ρ_r) which describes the reflected amount of light for a pair of directions on the same side of the surface. For transmitting materials like glass and water it becomes necessary to also describe the amount of transmitted light for a pair of directions on different sides of the surface. To refer to this type of interaction explicitly, the *Bidirectional Transmittance Distribution Function* (BTDF, ρ_t) is used. Finally, there is the *Bidirectional Scattering Distribution Function* (BSDF, ρ_s) which can be seen as the sum of BRDF and BTDF. Beyond that, the BSDF can be used to model the volumetric medium, too, since it does not depend on a surface and defines the amount of scattered light for any pair of directions.

The notations ρ_r, ρ_t and ρ_s will be used to distinguish the different types of scattering events. All three functions have the unit $[\mathrm{sr}^{-1}]$ and transform an incident *differential irradiance* into an excitant differential *radiance*

$$\rho = \mathrm{d}L(\boldsymbol{d}_\uparrow) / \mathrm{d}E(\boldsymbol{d}_\downarrow). \tag{II.34}$$

In cases where any of the three can be inserted, ρ will be used instead.

Any scattering function must fulfill the following properties to be physically plausible:

Def. Properties of BSDF

1. Nonnegative: $$\rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) \geq 0 \tag{II.35}$$

2. Reciprocal: $$\rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \rho(\boldsymbol{d}_\uparrow, \boldsymbol{d}_\downarrow) \tag{II.36}$$

3. Energy conserving: $$\int_\Omega \rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) \cos\theta_\uparrow \, \mathrm{d}\omega_\uparrow \leq 1 \tag{II.37}$$

Set of directions/*solid angle* Ω (Sec. II.1.1 p. 14)

Property two is required for the symmetry in the light transport algorithms. Otherwise using a different algorithm (tracing importance or photons) would result in different results. Some of our models have a flaw and violate this principle (e.g. *shading normals* in Section II.4.1). Thus, they need a special treatment.

There is also a physical property known as *Helmholtz-reciprocity* [Helmholtz 1867]. It says that any optical path, i.e. reflections and refractions, can be inverted. It does not hold for moving, magnetic and nonlinear media. Further, it does not apply to effects based on emission like luminescence or blackbody radiation. Those effects are not optical as they absorb and emit photons instead of bending the path of a photon. However, it can be said that any optical material enforces the reciprocity requirements while for the other we adopt it for practical reasons. For further reading of asymmetries also have a look in Veach's thesis [Veach 1997, pp. 135–194].

The property of energy conservation is enforced by physical laws and important to avoid an increase in energy by scattering events. The total

amount of reflected light can never be greater than the incoming light. On the other hand, it is valid to decrease the energy because absorption takes place. Moreover, in physics any material has some degree of absorption. Otherwise light could be trapped in a box and would stay there forever. This would cause an infinite bright *radiance* in our algorithms because they assume a constant supply of radiated energy from the light sources. I.e. by ignoring the time we would add more and more energy to the box with each indirection of light.

Nevertheless, we would like to have *energy preserving* models as well. That means energy should not vanish without explicit notion, in which case Equation (II.37) becomes an equality. Preferably, an albedo parameter should give explicit control over the amount of absorbed light.

II.5.1 ALBEDO

The albedo ϱ of a diffuse material is the percentage of reflected light with respect to the received radiation [Lambert 1760]. A surface with an albedo of 0.5 absorbs half of the energy in the respective spectrum. This idea can be generalized to any material as the average reflected energy assuming a constant illumination.

First, for a single outgoing (scattering) direction $\boldsymbol{d}$ we define the *directional albedo* as

$$\varrho(\boldsymbol{d}) = \int_\Omega \rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}) \, |\cos\theta_\downarrow| \, \mathrm{d}\omega_\downarrow \tag{II.38}$$

using a constant illumination of $L(\boldsymbol{d}_\downarrow) = 1$. Note that this definition is similar to the energy conservation, but the integration is done over the incoming light instead of over the reflected light. Since energy conservation and reciprocity hold, the albedo must be in $[0, 1]$, too. A value greater than one would cause an increase in energy in the reciprocal direction.

Energy conservation Eq. (II.37) p. 39

Taking the average over all possible scattering directions we get the *albedo*

$$\varrho = \frac{1}{|\Omega|} \int_\Omega \varrho(\boldsymbol{d}) \, \mathrm{d}\omega. \tag{II.39}$$

Energy conservation is violated if the *albedo* exceeds one. To see that, one can combine Equations (II.38), (II.39) and exchange the inner and outer integral. Then, the inner integral is equal to Equation (II.37) and should always be less than one. If that is the case, then the average due to the outer integral must be less than one, too. Now, if the average exceeds one, there must be at least a small set of directions which violates the energy conservation.

The other direction, that energy is conserved if $\varrho < 1$, is not true in general. The energy conservation can be violated for a small set of incident light directions, while the average albedo ϱ is still smaller than one.

II.5.2 REAL MATERIALS

In reality there are two fundamental types of materials: *conductors* and *dielectrics* (*nonconductors*). In between there is also the class of *semiconductor* materials which behave as *conductors* or *dielectrics* dependent on the environment conditions. In *conductors* the freely available charges may easily interact with the electromagnetic light. For perfect conductors this

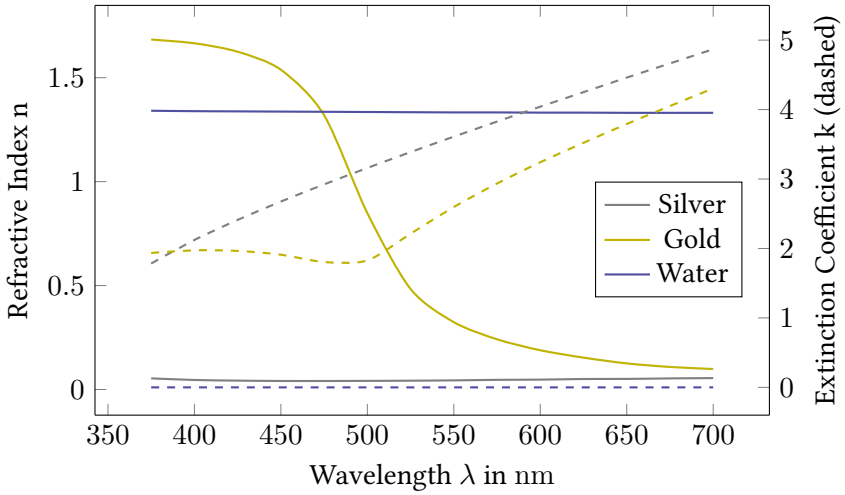


Figure II.12: Examples of refractive indices over the visible spectrum [Polyanskiy 2018]. While water seems to behave independent of the wavelength this is only the case within the visible spectrum. Outside, for example at $\lambda = 100\,\mu\text{m}$ its index is $n \approx 1.96, k \approx 0.53$. Indeed, all materials have varying refractive indices, although dielectrics are almost constant in the visible range.

causes an infinite refractive index η, producing a perfect reflector. In a real conductor the refractive index (and therefore reflectivity) changes over the spectrum causing a colored reflection. Figure II.12 shows some example refractive indices for different materials over the visible spectrum. It demonstrates that silver and water reflect light of all visible wavelengths evenly while gold shows a variation over this range. This causes white reflections for silver and water and a yellow one for gold.

For dielectric materials, like water, the refractive index is a real number $\eta = n$, although this might change under extreme circumstances or for wavelengths outside the visible spectrum. For conductors the refractive index is a complex value $\eta = n + \mathrm{i}k$ where the extinction coefficient k measures the absorption of light. Thereby, k is in the exponent in the transmittance function

Extinction / Attenuation

$$T = e^{-\int_0^\ell k(z)\,\mathrm{d}z} \tag{II.40}$$

known as *Beer-Lambert law* [Beer 1852; Lambert 1760]. It integrates the absorption over the size of a material sample ℓ. In case of uniform attenuation the law simplifies to $e^{-k\ell}$. Since the absorption in conductors is much higher than in dielectrics they only have a reflective component in most cases. Light is only transmitted in case of very thin layers.

The connection between refractive indices and the amount of reflected light is made by the Fresnel equations. Beyond the refractive index, the reflectance also depends on polarization and *magnetic permeability*, where the latter will not be handled here. For a visualization of the symbols and

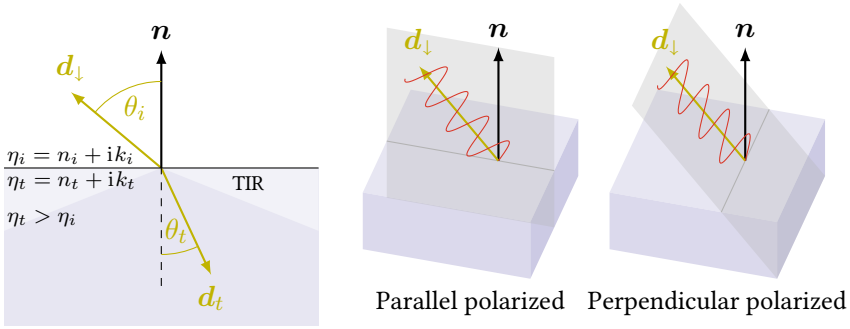


Figure II.13: Refraction of light at an interface. Total internal refraction (TIR) appears for directions inside the dense medium which do not have a counterpart on the other side. Right: polarization directions for $R_\parallel$ and $R_\perp$.

directions see Figure II.13. The Fresnel equations are

Fresnel equations

$$R_\parallel = \left(\frac{\eta_t \cos\theta_i - \eta_i \cos\theta_t}{\eta_t \cos\theta_i + \eta_i \cos\theta_t} \right)^2 \tag{II.41a}$$

$$R_\perp = \left(\frac{\eta_i \cos\theta_i - \eta_t \cos\theta_t}{\eta_i \cos\theta_i + \eta_t \cos\theta_t} \right)^2 \tag{II.41b}$$

$$F = \frac{R_\parallel + R_\perp}{2}, \tag{II.41c}$$

where $R_\parallel$ is the reflectance of parallel polarized light, $R_\perp$ the reflectance for perpendicular polarized light and F the reflectance of unpolarized light. The index i denotes quantities on the light-incident side whereas t is used for the side of transmitted light (i.e. the opposite side).

The angles θ_i and θ_t are related by Snell's law

$$\eta_i \sin\theta_i = \eta_t \sin\theta_t. \tag{II.42}$$

Snell's law

The angles themselves can and will become complex numbers for metals. In this case the imaginary part describes the phase shift of the wave, which we can ignore in computer graphics since the phase is not modeled anyway.

To be able to compute the Fresnel reflectance from the incident direction only, we can use Snell's law together with trigonometric Pythagoras ($\sin^2\theta + \cos^2\theta = 1$) and obtain

$$\cos\theta_t = \sqrt{1 - \frac{\eta_i^2}{\eta_t^2}(1 - \cos^2\theta_i)}. \tag{II.43}$$

In case of a nonmagnetic *dielectric-dielectric* interface for which $k_i = k_t = 0$ the Fresnel reflectances with respect to the incident angle are

$$R_\parallel = \left(\frac{n_t \cos\theta_i - n_i \sqrt{1 - (n_i/n_t)^2(1 - \cos^2\theta_i)}}{n_t \cos\theta_i + n_i \sqrt{1 - (n_i/n_t)^2(1 - \cos^2\theta_i)}} \right)^2 \tag{II.44a}$$

$$R_\perp = \left(\frac{n_i \cos\theta_i - n_t \sqrt{1 - (n_i/n_t)^2(1 - \cos^2\theta_i)}}{n_i \cos\theta_i + n_t \sqrt{1 - (n_i/n_t)^2(1 - \cos^2\theta_i)}} \right)^2 \tag{II.44b}$$

dielectric-dielectric interface, Combining Eqs. (II.41a), (II.41b) and (II.42)

which only uses the real parts n_i and n_t of the refraction index.

The second case which may appear in graphics is the *dielectric-conductor* interface. In this case $\eta_t = n_t + \mathrm{i}k_t$ is a complex number and the equations become

$$R_\perp = \frac{a^2 + b^2 - 2a\cos\theta_i + \cos^2\theta_i}{a^2 + b^2 + 2a\cos\theta_i + \cos^2\theta_i}$$
$$R_\parallel = R_\perp \frac{(a^2 + b^2)\cos^2\theta_i - 2a\cos\theta_i \sin^2\theta_i + \sin^4\theta_i}{(a^2 + b^2)\cos^2\theta_i + 2a\cos\theta_i \sin^2\theta_i + \sin^4\theta_i}$$

dielectric-conductor interface, Combining Eqs. (II.41a), (II.41b) and (II.42)

with

$$c = n_t^2 - k_t^2 - n_i^2 \sin^2\theta_i$$
$$a^2 + b^2 = \frac{1}{n_i^2}\sqrt{c^2 + 4n_t^2 k_t^2}$$
$$a^2 = \frac{1}{2n_i^2}\left(\sqrt{c^2 + 4n_t^2 k_t^2} + c\right)$$

These formulas can be found in Shirley's thesis [Shirley 1991, p. 15] and the PBRT book [Pharr et al. 2017, p. 520].

The cases for *conductor-dielectric* and *conductor-conductor* can be neglected due to the high absorption inside the conductor. Therefore, it is very unlikely to have incident light from the *conductor* side of an interface.

Using the above formulas, it is possible to compute the fraction of reflected light F, for which the excitant direction is the perfect reflected direction

$$\boldsymbol{d}_\uparrow = 2\langle \boldsymbol{d}_\downarrow, \boldsymbol{n}\rangle \boldsymbol{n} - \boldsymbol{d}_\downarrow. \tag{II.45}$$

Perfect reflection

Note that all vectors are oriented away from the surface which is a common convention. Using different notations leads to an inversion of signs for the associated vectors.

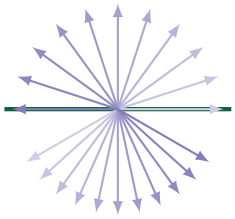
The transmitted light $(1 - F)$ is often treated differently for different models. For metals it is absorbed, for other opaque materials it will be scattered randomly and for transparent materials it will be refracted. The difference between a transparent and an opaque material is the amount of scattering. If scattering becomes very likely, the light will immediately be subject to another direction and spectrum changing interaction with the material. The computation of the refracted direction can be discarded in this case. For transparent materials the excitant direction is

$$\boldsymbol{d}_\uparrow = \left(\frac{\eta_i}{\eta_t}\langle \boldsymbol{d}_\downarrow, \boldsymbol{n}\rangle - \cos\theta_t\right)\boldsymbol{n} - \frac{\eta_i}{\eta_t}\boldsymbol{d}_\downarrow \tag{II.46}$$

Perfect refraction

where $\cos\theta_t$ is given by Equation (II.43).

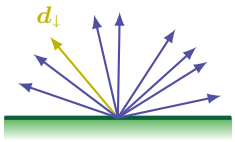
One last thing to notice is that *radiance* will change in a refraction. As visible in the side image all directions from an entire half-space will be squeezed into a smaller solid angle inside the denser medium. It can be shown [Pharr et al. 2017, pp. 526–527] that the radiance depends on the squared refraction indices



$$L_t = (1 - F)L_i \frac{\eta_t^2}{\eta_i^2}. \tag{II.47}$$

The reason is that *radiance* is a density per area and solid angle and that the solid angle is changed by the refraction with the given ratio.

II.5.3 LAMBERTIAN DIFFUSE BRDF



The simplest material model used in computer graphics is the Lambertian diffuse BRDF

$$\rho_{\mathrm{L}}(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \frac{\varrho}{\pi},$$

where $\varrho(\lambda) \in [0, 1]$ is the constant reflectance with values smaller than one modeling the absorption.

It is independent of the incident and excitant direction which often allows simplifications in light transport algorithms. With respect to real materials it can be seen as maximum opaque material with a perfect uniform scattering. A different perspective is that of an area light source (Section II.3.2) with the *radiance* $L_\uparrow = \frac{\varrho}{\pi}\,\mathrm{d}E$. This also gives us the importance sampling PDF $p(\boldsymbol{d}_\uparrow)$ and the sampling algorithm `sampleCosineDir`. Like for area lights, the *intensity* is cosine-distributed and it is more effective to sample this distribution than to sample uniform outgoing directions.

$p_{\cos}(\boldsymbol{d}_\uparrow) = \cos\theta_\uparrow/\pi$ p. 32
sampleCosineDir p. 32

II.5.4 OREN-NAYAR DIFFUSE BRDF

The Lambertian diffuse BRDF is a poor representation for real rough surfaces like the moon or paper. As a solution, Oren and Nayar [1994] modeled the surface of rough objects with Lambertian microfacets. Between those microfacets shadowing and multiple scattering may occur, resulting in a more realistic appearance. They derived two models: a complex one which lacks a closed form solution and a simpler, fitted approximation known as the *qualitative model*.

The *qualitative model* is

$$\rho_{\mathrm{ON}}(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \frac{\varrho}{\pi}\left(A + B \cdot (\cos(\phi_\downarrow - \phi_\uparrow))^+ \cdot \sin\theta_{\max} \cdot \tan\theta_{\min}\right) \quad \text{(II.48)}$$

$$\begin{aligned} A &= 1 - \frac{\alpha^2}{2\alpha^2 + 0.66} \\ B &= \frac{0.45\alpha^2}{\alpha^2 + 0.09} \\ \theta_{\max} &= \max(\theta_\downarrow, \theta_\uparrow) \\ \theta_{\min} &= \min(\theta_\downarrow, \theta_\uparrow) \end{aligned}$$

where α is a roughness parameter in radians with a valid range of $[0, \pi]$. By applying several trigonometric identities

$$\begin{aligned} \tan\theta_{\min} &= \frac{\sin\theta_{\min}}{\cos\theta_{\min}} \\ \cos(\phi_\downarrow - \phi_\uparrow) &= \cos\phi_\downarrow \cdot \cos\phi_\uparrow + \sin\phi_\downarrow \cdot \sin\phi_\uparrow \\ \cos\min(\theta_\downarrow, \theta_\uparrow) &= \max(\cos\theta_\downarrow, \cos\theta_\uparrow) \end{aligned}$$

it is possible to express the entire evaluation with the vectors directly without the use of trigonometric functions. The last line is only valid for the restricted domain $[0, \pi]$ which is always the case for the θ angles which lie in $[0, \pi/2]$.

```
# Assume that A, B and ϱ are computed upfront.
# Further, d↓.z>0 and d↑.z>0 (local tangent space).
func evalOrenNayar(d↓, d↑) -> ρr
  # Get all the trigonometric quantities
  cosTi = d↓.z
  cosTo = d↑.z
  sinTi = sqrt(1 - cosTi^2)
  sinTo = sqrt(1 - cosTo^2)
  # The +ε prevents the division by 0
  cosPi = d↓.x / (sinTi + ε)
  sinPi = d↓.y / (sinTi + ε)
  cosPo = d↑.x / (sinTo + ε)
  sinPo = d↑.y / (sinTo + ε)
  # Evaluate the model
  cosPiMinusPo = max(0, cosPi * cosPo + sinPi * sinPo)
  tmp = A + B * cosPiMinusPo * sinTi * sinTo / max(cosTi, cosTo)
  return ϱ / π * tmp * cosTo
end
```

SAMPLING

Often the Oren-Nayar model is sampled using the same routine like for Lambertian diffuse surfaces. This yields an optimal choice for the A term, but does not follow the B part. Still, using the cosine sampling distribution is never a terrible choice. Even for the high roughness $\alpha = \pi$, the BRDF is still dominated by the Lambertian part ($A \approx 0.516 > B \approx 0.446$). I could not find any better importance sampling method for the Oren-Nayar model in the literature, so the following is likely a new contribution.

To sample the entire function from Equation (II.48) we can apply multiple importance sampling. For brevity, the excitant sizes $\theta_\uparrow, \phi_\uparrow$ will be written θ, ϕ and $\phi_\downarrow - \phi_\uparrow$ will be replaced with $\Delta\phi$ in the following.

Multiple Importance Sampling II.2.3 p. 22

With a probability of P_A we chose the `sampleCosineDir` to sample the A part, where a good value of P_A will be introduced later. Otherwise, we would like to sample the part $B \cdot \cos\Delta\phi \cdot \sin\theta_\downarrow \cdot \sin\theta / \cos\theta_{\text{min}}$ multiplied with $\cos\theta$ from the irradiance projection. We can write $\sin\theta_\downarrow \cdot \sin\theta$ instead of $\sin\theta_{\text{max}} \cdot \sin\theta_{\text{min}}$ after using the trigonometric identity of the $\tan\theta_{\text{min}}$ term, since both terms appear regardless of the case. Then, B and $\sin\theta_\downarrow$ can be removed from the derivation of a sampling density since both only scale the model and will be removed in the normalization process anyway. Due to the still required $\cos\theta_{\text{min}}$ we get two different cases for the sampling distribution

sampleCosineDir p. 32

Differential irradiance Eq. (II.5) p. 16

$$p(\theta, \Delta\phi) \propto f(\theta, \Delta\phi) = \begin{cases} f_1 : \cos\Delta\phi \cdot \sin\theta \cdot \frac{\cos\theta}{\cos\theta_\downarrow} & \text{if } \theta_\downarrow < \theta \\ f_2 : \cos\Delta\phi \cdot \sin\theta. \end{cases}$$

In the second case the $\cos\theta_{\text{min}}$ equals $\cos\theta$ and cancels out with the outgoing cosine from the irradiance projection to the surface.

Since f is defined piecewise, we cannot apply the inverse CDF method directly. Instead, we have to select one of two inverse CDFs in the end. Integrating over each of the two cases of f individually gives

Inverse CDF II.2.5 p. 25

$$F_1(\theta, \Delta\phi) = (\sin\Delta\phi + 1)\frac{\sin^3\theta}{3\cos\theta_\downarrow}$$
$$F_2(\theta, \Delta\phi) = (\sin\Delta\phi + 1)\frac{2\theta - \sin(2\theta)}{4}$$

where F_2 has no closed form inverse for θ. It would be possible to solve the inversion by an iterative method. However, this would increase the costs for this model too far. Instead I used the approximation

$$F_2(\theta, \Delta\phi) \approx (\sin\Delta\phi + 1)\left(\frac{2\theta}{\pi}\right)^{2.4}. \tag{II.49}$$

In the next step the two functions and their integrals must be normalized to obtain valid PDFs and CDFs. The domain of $\Delta\phi$ is $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and not the entire circle due to the $()^+$ in Equation (II.48). Thus, we have $\frac{\pi}{2}$ as the maximum for both sampling dimensions and get the normalized functions

$$\hat{F}_1(\theta, \Delta\phi) = \frac{F_1(\theta, \Delta\phi)}{F_1(\frac{\pi}{2}, \frac{\pi}{2})} = \frac{1}{2}(\sin\Delta\phi + 1)\cdot\sin^3\theta, \tag{II.50a}$$

$$p_1(\theta, \Delta\phi) = \frac{f_1(\theta, \Delta\phi)}{F_1(\frac{\pi}{2}, \frac{\pi}{2})} = \frac{3}{2}\cos\Delta\phi\cdot\sin\theta\cdot\cos\theta, \tag{II.50b}$$

$$\hat{F}_2(\theta, \Delta\phi) = \frac{F_2(\theta, \Delta\phi)}{F_2(\frac{\pi}{2}, \frac{\pi}{2})} = \frac{1}{2}(\sin\Delta\phi + 1)\cdot\left(\frac{2\theta}{\pi}\right)^{2.4}, \tag{II.50c}$$

$$p_2(\theta, \Delta\phi) = \frac{f_2(\theta, \Delta\phi)}{F_2(\frac{\pi}{2}, \frac{\pi}{2})} = \frac{0.4059696256\cdot\cos\Delta\phi\cdot\theta^{1.4}}{\sin\theta}. \tag{II.50d}$$

Finally, the inverse CDF method can be applied in the two steps $\mathcal{U}_1 = F(\theta, \frac{\pi}{2})$ and $\mathcal{U}_2 = F(\theta, \Delta\phi)/F(\theta, \frac{\pi}{2})$: 2D inverse CDF II.2.5 p. 26

$$\begin{aligned} \theta_1 &= \arcsin\left(\mathcal{U}_1^{1/3}\right) & \theta_2 &= \frac{\pi}{2}\cdot\mathcal{U}_1^{1/2.4} \\ \Delta\phi_1 &= \arcsin(2\mathcal{U}_2 - 1) & \Delta\phi_2 &= \arcsin(2\mathcal{U}_2 - 1) \end{aligned}$$

which lead to the final sampling direction

$$\boldsymbol{d}_\uparrow = \begin{bmatrix}\sin\theta\cdot\cos(\phi_\downarrow + \Delta\phi)\\ \sin\theta\cdot\sin(\phi_\downarrow + \Delta\phi)\\ \cos\theta\end{bmatrix} = \begin{bmatrix}\sin\theta\cdot(\cos\phi_\downarrow\cos\Delta\phi - \sin\phi_\downarrow\sin\Delta\phi)\\ \sin\theta\cdot(\sin\phi_\downarrow\cos\Delta\phi + \cos\phi_\downarrow\sin\Delta\phi)\\ \cos\theta\end{bmatrix}.$$

This leaves us with the question which of the three sampling alternatives (Lambert, p_1, p_2) we have to sample how often. To match the target function well, P_A and P_B should be proportional to the weights of the layer. These weights are defined by the constant factors A and B as well as the incident sizes $\cos\theta_\downarrow$ and $\sin\theta_\downarrow$. A still open problem is the division by $\cos\theta_{\text{min}}$ which depends on the unknown outgoing value. Experimentally, $P_A \propto A$ and $P_B \propto B\cdot\sin\theta_\downarrow$ showed the best results, leading to $P_A = A/(A + B\cdot\sin\theta_\downarrow)$. Additionally dividing P_B by $\cos\theta_\downarrow$ leads to a useless sampler with extreme throughput values. Thus, the full PDF becomes Throughput Sec. II.2.2 p. 20

$$p(\theta, \phi) = \frac{1}{A + B\sin\theta_\downarrow}\left(A\cdot p_{\cos}(\theta, \phi) + B\sin\theta_\downarrow\cdot p_{\{1,2\}}(\theta, \phi_\downarrow - \phi)\right) \tag{II.51}$$

$p_{\cos}(\boldsymbol{d}_\uparrow) = \cos\theta_\uparrow/\pi$ p. 32

where both choices of the PDF for the second term should lead to a better sampling scheme than the cosine sampling only.

Figure II.14 shows the throughput values for the Oren-Nayar BRDF at a flat incident direction over all excitant directions. All other incident directions are less extreme and all three sampling methods become identical for

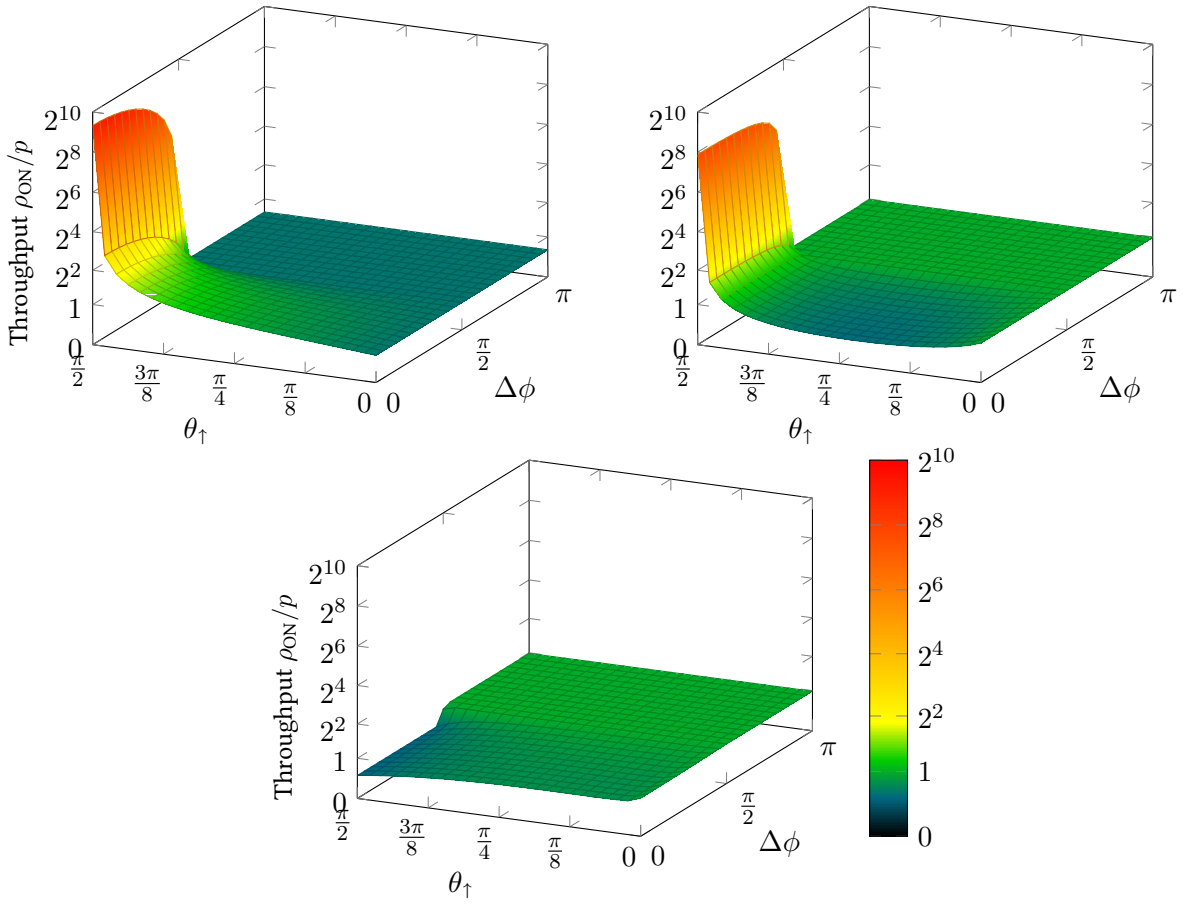


Figure II.14: Throughput values of: pure cosine importance sampling (top left), Equation (II.51) with p_1 (top right) and (II.51) with p_2 (bottom) for the flat incident direction $\theta_\downarrow = 1.57\,\mathrm{rad}$. The values are most extremal at these grazing angles. Values below one are preferred, whereas large values may lead to fireflies.

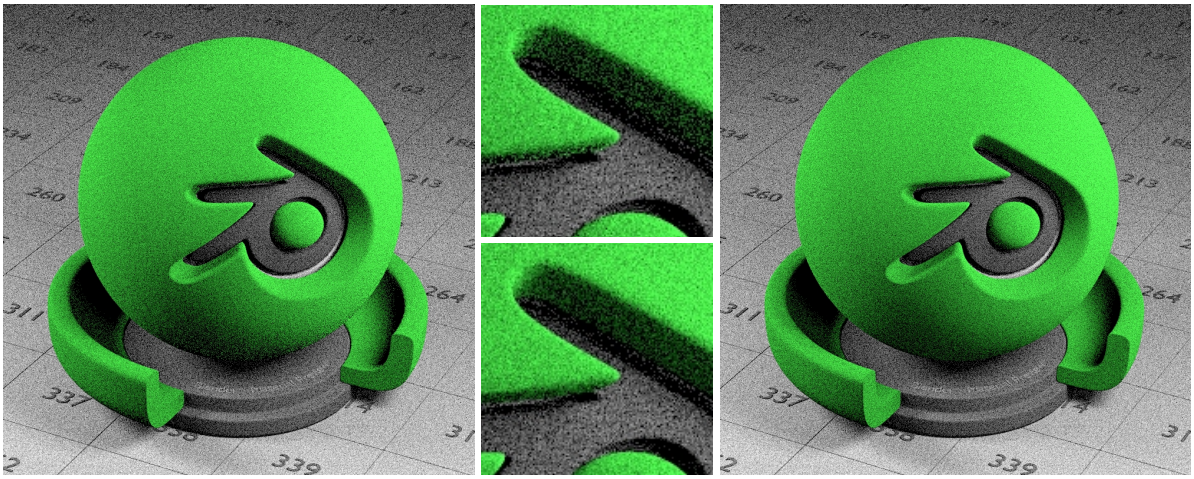


Figure II.15: Sampled direct illumination (400spp, no next event estimation) with cosine sampling (left/top) and the PDF (II.51) using p_2 (Equation (II.50d)) (right/bottom).

normal incidence. Using p_1 decreases the maximum value by $\approx 2\times$, but is still a bad sampler. On the other side, p_2 oversamples the same region which is much better for avoiding extreme variance values. Therefore, p_2 is used in the final sampling routine given below. Concluding, Figure II.15 demonstrates that the newly proposed sampling technique increases the robustness in a practical test.

p_1 Eq. (II.50b) p. 46
p_2 Eq. (II.50d) p. 46

Novel importance sampling of the Oren-Nayar model

```
func sampleOrenNayarDir(d↓, A, B, U1, U2, U3) -> d, ρr, p
  # Compute incident sizes
  cosThetaI = abs(d↓.z)
  sinThetaI = sqrt(1 - cosThetaI * cosThetaI)
  sinPhiI = if sinThetaI > 1e-4: d↓.x / sinThetaI else 0 end
  cosPhiI = if sinThetaI > 1e-4: d↓.y / sinThetaI else 1 end
  # Decide between Cosine and second sampling term
  PA = A / (A + B * sinThetaI)
  if U3 < PA: # cosine sampling
    cosThetaO = sqrt(U1)
    sinThetaO = sqrt(1 - U1)
    phi = U2 * 2 * π
    sinPhiO, cosPhiO = sincos(phi)
    # for PDF/BRDF: cos(φi-φo) = cos(φi)cos(φo) + sin(φi)sin(φo)
    cosDeltaPhi = max(0, cosPhiI * cosPhiO + sinPhiI * sinPhiO)
  else # B part sampling
    thetaO = π / 2 * pow(U1, 1 / 2.4)
    sinThetaO, cosThetaO = sincos(thetaO)
    sinDeltaPhi = 2 * U2 - 1
    cosDeltaPhi = sqrt(1 - sinDeltaPhi * sinDeltaPhi)
    # Transform local Δφ into tangent space cosφ and sinφ.
    cosPhiO = cosPhiI * cosDeltaPhi - sinPhiI * sinDeltaPhi
    sinPhiO = sinPhiI * cosDeltaPhi + cosPhiI * sinDeltaPhi
  end
  # Compute final PDF
  thetaExp = pow(acos(cosThetaO), 1.4)
  pdfA = cosThetaO / π
  pdfB = 0.40596962562901 * cosDeltaPhi * thetaExp / sinThetaO
  pdf = PA * pdfA + (1-PA) * pdfB
  # Compute BRDF
  cosMax = max(cosThetaI, cosThetaO)
  ρr = (A + B * cosDeltaPhi * sinThetaI * sinThetaO / cosMax) / π
  return [sinThetaO*sinPhiO, sinThetaO*cosPhiO, cosThetaO], ρr, pdf
end
```

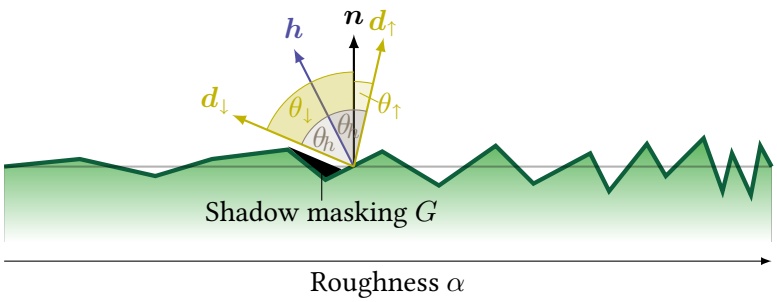


Figure II.16: A microfacet surface

II.5.5 MICROFACET BRDFS

Purely *specular* materials (e.g. water or glass) are characterized by the Fresnel reflectance and the two equations (II.45) and (II.46). For one incident direction they split the path into two deterministic components: reflection and refraction. However, most materials are not as smooth and have a rough surface. These *glossy* materials reflect and refract light in a broader distribution.

Reflection Eq. (II.45) p. 43, Refraction Eq. (II.46) p. 43

Beckmann and Spizzichino [1963] introduced the microfacet model which explains *glossy* surfaces by a high number of *specular* Fresnel reflecting facets on a microscopic level. By defining a distribution of normals D of the facets one can parametrize the macroscopic appearance of the surface.

Def. Distribution of microfacets D

The most used model in computer graphics is the Torrance-Sparrow model [Torrance and Sparrow 1967]

$$\rho_{\mathrm{T}}(\boldsymbol{d}_{\downarrow}, \boldsymbol{d}_{\uparrow}) = \frac{D(\boldsymbol{h}) \cdot G(\boldsymbol{d}_{\downarrow}, \boldsymbol{d}_{\uparrow}) \cdot F(\boldsymbol{d}_{\downarrow}, \boldsymbol{h})}{4\,|\langle \boldsymbol{d}_{\downarrow}, \boldsymbol{n}\rangle \cdot \langle \boldsymbol{d}_{\uparrow}, \boldsymbol{n}\rangle|}, \tag{II.52}$$

where $\boldsymbol{h}$ is the *half-vector* which is the normal of the microfacet for which $\boldsymbol{d}_{\uparrow}$ is the reflected direction of $\boldsymbol{d}_{\downarrow}$. It is calculated by

Reflection Eq. (II.45) p. 43

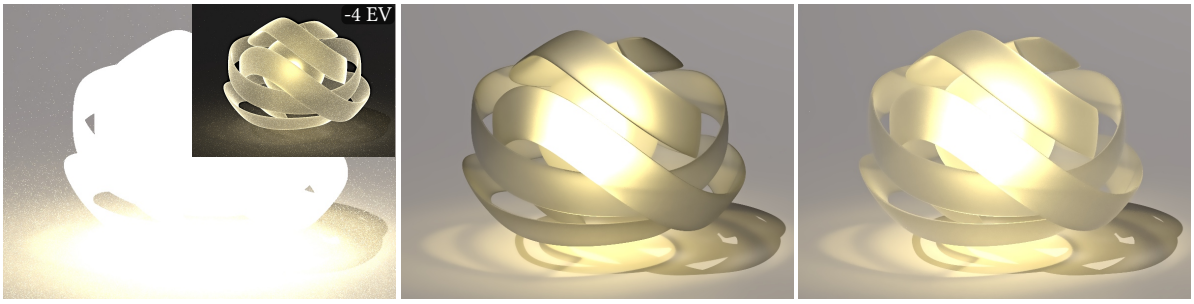
$$\boldsymbol{h} = \frac{\boldsymbol{d}_{\downarrow} + \boldsymbol{d}_{\uparrow}}{\|\boldsymbol{d}_{\downarrow} + \boldsymbol{d}_{\uparrow}\|}. \tag{II.53}$$

Def. Half-vector

Also, see Figure II.16 for the notation. The distribution of microfacets D determines the amount of visible surfaces into the *half-vector* direction $\boldsymbol{h}$. As visible in the image, this amount may vary for certain pairs of incident and excitant directions because of shadowing. Therefore, the shadow masking function G, which depends on the distribution of microfacets, accounts for this visibility. Unfortunately, masking leads to an energy loss at flat incident angles for rough surfaces (Figure II.17 (b)). In reality the ray would be reflected multiple times between the facets instead of being discarded as modeled by G (Figure II.17 (c)). More details and solutions are described in Section II.5.8. Figure II.17 (a) also demonstrates what happens if the geometry term is discarded, in which case the result is physically implausible. Particularly, the energy conservation does not hold in this case, causing a severe increase in brightness.

Energy loss Sec. II.5.8 p. 55

There are two common models to describe the amount of shadowing. The Smith [1967] model assumes that the orientation of normals is independent of the probability of masking which leads to a separable function



(a) $G = 1$ (b) V-cavity G (Eq. (II.54)) (c) Multiple-scattering

Figure II.17: Influence of microfacet shadow masking function G for a microfacet BTDF ([Walter et al. 2007]).

$G(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = G_1(\boldsymbol{d}_\downarrow) \cdot G_1(\boldsymbol{d}_\uparrow)$. Thereby, the function G_1 depends on the distribution of microfacets D and must be derived anew for each distribution.

Smith (uncorrelated)

V-cavity

The conceptually simpler V-cavity model [Torrance and Sparrow 1967] assumes that the surface is composed of many V-shaped cavities for which the shadowing can be computed without knowing the distribution D. According to Heitz [2014] both models are physically plausible, where the Smith model yields the slightly more realistic results. The work of Heitz [2014] also gives a good introduction to the two mentioned and further shadow masking terms.

For reasons of simplicity I will stick to the V-cavity model here. Therefore, the geometry shadowing term is

$$G(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow, \boldsymbol{h}, \boldsymbol{n}) = \min\left(G_1(\boldsymbol{d}_\downarrow, \boldsymbol{h}, \boldsymbol{n}), G_1(\boldsymbol{d}_\uparrow, \boldsymbol{h}, \boldsymbol{n})\right) \tag{II.54}$$

V-cavity shadow masking function

$$G_1(\boldsymbol{d}, \boldsymbol{h}, \boldsymbol{n}) = \begin{cases} 0 & \text{if } \langle \boldsymbol{d}, \boldsymbol{n} \rangle \cdot \langle \boldsymbol{d}, \boldsymbol{h} \rangle < 0 \\ \min\left(1, 2 \left| \frac{\langle \boldsymbol{d}, \boldsymbol{n} \rangle \cdot \langle \boldsymbol{h}, \boldsymbol{n} \rangle}{\langle \boldsymbol{d}, \boldsymbol{h} \rangle} \right| \right) & \text{else} \end{cases}$$

The case with < 0 masks those combinations of *half-vector* and normal which are impossible because they are visible from different sides. The problem is the same as before for shading normals (see Figure II.10 p. 36).

Next, the distribution function must be chosen. A widely spread choice is that of the GGX function [Trowbridge and Reitz 1975; Walter et al. 2007]

$$D_{\text{GGX}}(\boldsymbol{h}, \boldsymbol{n}) = \frac{1}{\pi \alpha^2 \left(\langle \boldsymbol{h}, \boldsymbol{n} \rangle^2 + \frac{1 - \langle \boldsymbol{h}, \boldsymbol{n} \rangle^2}{\alpha^2} \right)^2}, \tag{II.55a}$$

GGX normal distribution

which also has an anisotropic variant [Heitz and d'Eon 2014]

$$D_{\text{aGGX}}(\boldsymbol{h}, \boldsymbol{n}) = \frac{1}{\pi \alpha_x \alpha_y \left(\langle \boldsymbol{h}, \boldsymbol{n} \rangle^2 + \frac{\langle \boldsymbol{h}, \boldsymbol{x} \rangle^2}{\alpha_x^2} + \frac{\langle \boldsymbol{h}, \boldsymbol{y} \rangle^2}{\alpha_y^2} \right)^2}. \tag{II.55b}$$

Here, $\boldsymbol{x}$ and $\boldsymbol{y}$ are two tangential vectors of the surface (perpendicular to $\boldsymbol{n}$). They are often defined by the texture coordinates and do not need to be mutually perpendicular ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$ is valid) but must be linearly independent.

Another common choice is the Beckmann-Spizzichino model [Beckmann and Spizzichino 1963]

$$D_{\text{BS}}(\boldsymbol{h},\boldsymbol{n}) = \frac{\exp\left(-\frac{\tan^2\theta_{\boldsymbol{h}}}{\alpha^2}\right)}{\pi\alpha^2\left\langle\boldsymbol{h},\boldsymbol{n}\right\rangle^4} = \frac{\exp\left(\frac{\langle\boldsymbol{h},\boldsymbol{n}\rangle^2-1}{\alpha^2\langle\boldsymbol{h},\boldsymbol{n}\rangle^2}\right)}{\pi\alpha^2\left\langle\boldsymbol{h},\boldsymbol{n}\right\rangle^4} \tag{II.56a}$$

BS normal distribution

for which Heitz and d'Eon [2014] also developed an anisotropic variant

$$D_{\text{aBS}}(\boldsymbol{h},\boldsymbol{n}) = \frac{\exp\left(-\frac{\langle\boldsymbol{h},\boldsymbol{x}\rangle^2}{\alpha_x^2}-\frac{\langle\boldsymbol{h},\boldsymbol{y}\rangle^2}{\alpha_y^2}\right)}{\pi\alpha_x\alpha_y} \tag{II.56b}$$

And finally there is the Blinn-Phong NDF also known as cosine lobe

$$D_{\text{Cos}} = \frac{n+2}{2\pi}\left\langle\boldsymbol{h},\boldsymbol{n}\right\rangle^n = \frac{1}{\pi\alpha^2}\left\langle\boldsymbol{h},\boldsymbol{n}\right\rangle^{2/\alpha^2-2}. \tag{II.57}$$

Cosine normal distribution

Usually the Blinn-Phong model is parametrized by an exponent n. The second form is obtained by setting the normalization factor $(n+2)/2\pi$ equal to $1/\pi\alpha^2$ such that we can use the unified parameter α.

Finally, there is a choice for the reflectance F in Equation (II.52). For realistic materials this is the Fresnel reflectance from Equation (II.41c) p. 42. A different model is Schlick's approximation [1993]

$$F_{\text{Schlick}}(\boldsymbol{d}_\downarrow,\boldsymbol{h}) = F_0 + (1-F_0)(1-\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}\rangle)^5$$
$$\text{with}\quad F_0 = \left(\frac{n_i-n_t}{n_i+n_t}\right)^2.$$

However, this approximation causes two problems in physical based rendering. First, the function will approach one only for $\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}\rangle = 0$ independent of the incident direction. This ignores the angle of total internal reflection in the dense medium, for which the reflectivity should approach one earlier. The second problem is the missing reciprocity in case of transmission. For reflections the term will evaluate to the same number for swapped directions. It will not be equal for the refraction *half-vector* $\boldsymbol{h}_t$ (Equation (II.60) in the next section). Therefore, Schlick's approximation should not be used for bidirectional light transport algorithms.

SAMPLING

As usual, the sampled distribution should match $\rho_{\text{T}}(\boldsymbol{d}_\downarrow,\boldsymbol{d}_\uparrow)\cdot\langle\boldsymbol{d}_\uparrow,\boldsymbol{n}\rangle$ to achieve a low variance in Monte Carlo sampling. First, we factor out the Fresnel term from Equation (II.52). It will be handled in Section II.5.7 on multilayer materials because different combinations need special treatments. This leaves us the distribution $D\cdot G/(4\left\langle\boldsymbol{d}_\downarrow,\boldsymbol{n}\right\rangle)$ which cannot be sampled directly with the inverse CDF method.

$\rho_{\text{T}}(\boldsymbol{d}_\downarrow,\boldsymbol{d}_\uparrow)$ from Eq. (II.52) p. 49

Instead we can sample a *half-vector* following D and reflect the incident direction using Equation (II.45). This, however, ignores the geometric shadowing in the sampling process. In consequence it is possible to choose invalid *half-vectors* which will cause zero contribution samples. To overcome this issue it is possible to sample the distribution of visible normals

$$D_\perp(\boldsymbol{d}_\downarrow,\boldsymbol{h},\boldsymbol{n}) = \frac{G_1(\boldsymbol{d}_\downarrow,\boldsymbol{h},\boldsymbol{n})\cdot|\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}\rangle|\cdot D(\boldsymbol{h},\boldsymbol{n})}{|\langle\boldsymbol{d}_\downarrow,\boldsymbol{n}\rangle|}$$

Def. Distribution of visible normals $D_\perp$

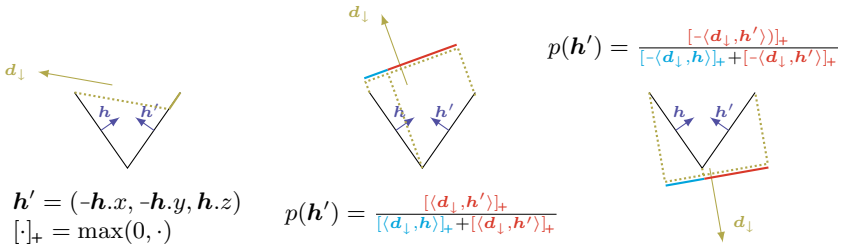


Figure II.18: For a chosen cavity either the primary *half-vector* $\boldsymbol{h}$ or its adjoint $\boldsymbol{h}'$ or both are visible to an incident ray. By choosing $\boldsymbol{h}'$ randomly with $p(\boldsymbol{h}')$ proportional to the visible area, the distribution of visible normals is sampled. This also applies if the surface is hit from below (right).

as shown in [Heitz and d'Eon 2014]. It guarantees that the incident direction is consistent with respect to normal $\boldsymbol{n}$ and *half-vector* $\boldsymbol{h}$ while it is still possible to have a shadowed excitant direction.

$D_\perp$ is not equal to the probability density of the outgoing direction, because the reflection operator transforms this distribution again. To compensate the change in a distribution it must be multiplied with the Jacobian of the operator (e.g. see [Pharr et al. 2017, pp. 812–813] for more details)

$$\left\| \frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}\boldsymbol{d}_\uparrow} \right\| = \frac{1}{4\,|\langle \boldsymbol{d}_\downarrow, \boldsymbol{h} \rangle|}.$$

Therefore, the distribution of the sampled excitant direction is

$$p(\boldsymbol{d}_\uparrow) = \frac{G_1(\boldsymbol{d}_\downarrow, \boldsymbol{h}, \boldsymbol{n}) \cdot D(\boldsymbol{h}, \boldsymbol{n})}{4\,|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n} \rangle|} \tag{II.58}$$

Def. Distribution of sampled directions for microfacet reflections

which still ignores part of the geometric shadowing, but is fairly close to the desired shape.

Heitz and d'Eon [2014] derived sampling algorithms for $D_\perp$ with respect to the Smith and the V-cavity model. For the V-cavity model the trick is to randomly choose the adjoint *half-vector* $\boldsymbol{h}'$ dependent on the visibility of primary and adjoint *half-vector*. Figure II.18 visualizes the swap for a chosen cavity. Since this process is independent of the distribution $D(\boldsymbol{h})$ we can first sample a primary half vector for the anisotropic GGX (or a different) distribution with the algorithm (See supplemental 2 from [Heitz and d'Eon 2014])

Sampling of anisotropic GGX $D_{\mathrm{GGX}}(\boldsymbol{h}, \boldsymbol{n})\,\langle \boldsymbol{h}, \boldsymbol{n} \rangle$

```
func sampleGGX(U1, U2) -> [h, p]
  s = sqrt(U1 / (1 - U1))
  slopeX = s * cos(2 * π * U2)
  slopeY = s * sin(2 * π * U2)
  h = normalize([-αx * slopeX, -αy * slopeY, 1])
  # The half-vector pdf can be expressed with respect to the known
  # PDF of slopes and h.z (cos between half-vector and normal).
  tmp = slopeX^2 + slopeY^2 + 1
  return [h, 1 / (π * αx * αy * tmp^2 * h.z^3)]
end
```

Sampling of anisotropic Beckmann-Spizzichino $D_{\mathrm{B}}(\boldsymbol{h}, \boldsymbol{n})\,\langle \boldsymbol{h}, \boldsymbol{n} \rangle$

```
func sampleBeckmannSpizzichino(U1, U2) -> [h, p]
  s = sqrt(-log(1 - U1)) # (1 - U1) ∈ (0, 1]
  slopeX = s * cos(2 * π * U2)
  slopeY = s * sin(2 * π * U2)
  h = normalize([-αx * slopeX, -αy * slopeY, 1])
  # Inserting slopeX/Y into the pdf reverts the sqrt and log
  # functions -> simple PDF of slopes.
```

```
  return [h, (1 - 𝒰1) / (π * αx * αy * h.z^3)]
end
```

Finally, the following algorithm can be used to choose the *half-vector* proportional to its visibility

```
func choseVisibleHV(d↓, h0, 𝒰3) -> h
  # Adjoint normal
  h1 = [-h0.x, -h0.y, h0.z]
  # Clamp to 0 if invisible
  iDotH0 = max(0, dot(d↓, h0) * sign(d↓.z))
  iDotH1 = max(0, dot(d↓, h1) * sign(d↓.z))
  if 𝒰3 < iDotH1 / (iDotH0 + iDotH1):
    return h1 # take adjoint half-vector
  else
    return h0 # keep original
end
```

Sampling $D_\perp(\boldsymbol{d}_\downarrow, \boldsymbol{h}, \boldsymbol{n})$ given $\boldsymbol{h}0$ with distribution $D(\boldsymbol{h}0, \boldsymbol{n})\,\langle \boldsymbol{h}, \boldsymbol{n}\rangle$

Note, that the incident direction in local space $\boldsymbol{d}_\downarrow$. z is negative if the ray arrived from the inner side of the interface. This is possible for transparent objects like water. Consequently, we need the visibility of the cavity as seen from below and have to switch the signs in lines 5 and 6.

II.5.6 MICROFACET BTDFS

The Torrance-Sparrow model is capable of modeling the reflectance of a rough surface. Analogously, Walter's model [Walter et al. 2007] complements the model with the refractive component. It is based on the same concept of microscopic Fresnel-facets. However, due to the higher complexity of the refraction operator more cosine terms remain in the final form

$$\rho_{\mathrm{W}}(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \frac{D(\boldsymbol{h}_t)\cdot G(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)\cdot (1 - F(\boldsymbol{d}_\downarrow, \boldsymbol{h}_t))}{|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n}\rangle \cdot \langle \boldsymbol{d}_\uparrow, \boldsymbol{n}\rangle|} \frac{\eta_t^2\,|\langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t\rangle \cdot \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t\rangle|}{(\eta_i\,\langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t\rangle + \eta_t\,\langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t\rangle)^2} \tag{II.59}$$

Refraction indices η_i, η_t from Sec. II.5.2

where $\boldsymbol{h}_t$ is the refraction half vector which differs from the reflection half vector:

$$\boldsymbol{h}_t = -\frac{\eta_i \boldsymbol{d}_\downarrow + \eta_t \boldsymbol{d}_\uparrow}{\|\eta_i \boldsymbol{d}_\downarrow + \eta_t \boldsymbol{d}_\uparrow\|}. \tag{II.60}$$

Reflection half-vector Eq. (II.53) p. 49

$$\boldsymbol{h} = \frac{\boldsymbol{d}_\downarrow + \boldsymbol{d}_\uparrow}{\|\boldsymbol{d}_\downarrow + \boldsymbol{d}_\uparrow\|}$$

All remaining terms D, G and F are the same as in the Torrance-Sparrow model.

It turns out that the refractive BTDFs ρ_t do not satisfy the reciprocity condition including Walter's model ρ_{W}. Instead, the BTDF satisfies

$$\eta_i^2 \rho_t(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \eta_t^2 \rho_t(\boldsymbol{d}_\uparrow, \boldsymbol{d}_\downarrow).$$

The reason is that the directions in the hemisphere of the less dense medium are compressed into a smaller solid angle on the dense side. As mentioned before, the *radiance*, which depends on angular density, is scaled by Equation (II.47) causing the above behavior of the BTDF.

This asymmetry due to refractions is well known and details can be found in [Pharr et al. 2017, p. 961] and [Veach 1997, pp. 139–143, 171]. As a solution, the adjoint BTDF is defined as

$$\rho_t^*(\boldsymbol{d}_\uparrow, \boldsymbol{d}_\downarrow) = \frac{\eta_i^2}{\eta_t^2} \rho_t(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)$$

and used for evaluations on the light sub-path. In practice this requires the tracking of the current transported quantity (*radiance*/*importance*). Note that if a path leaves the medium again, the refraction index of the medium cancels out. Visible differences in brightness only appear, if the sensor and the light source are within two different media. Then the total *radiance* is scaled by the ratio of the refraction indices of the source and the target media.

SAMPLING

To sample the microfacet transmittance from Equation (II.59) we can apply the same procedure as previously described. First, the visible half vector $\boldsymbol{h}_t$ is sampled. Then a refracted direction is computed using Equation (II.46). Applying the Jacobian of the refraction operator

$$\left\| \frac{\mathrm{d}\boldsymbol{h}_t}{\mathrm{d}\boldsymbol{d}_\uparrow} \right\| = \frac{\eta_t^2 \left| \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t \rangle \right|}{\left(\eta_i \langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t \rangle + \eta_t \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t \rangle \right)^2}$$

the final sampled PDF is

$$p(\boldsymbol{d}_\uparrow) = \frac{D(\boldsymbol{h}_t, \boldsymbol{n}) \cdot G_1(\boldsymbol{d}_\downarrow, \boldsymbol{h}_t, \boldsymbol{n})}{\left| \langle \boldsymbol{d}_\downarrow, \boldsymbol{n} \rangle \right|} \frac{\eta_t^2 \left| \langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t \rangle \cdot \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t \rangle \right|}{\left(\eta_i \langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t \rangle + \eta_t \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t \rangle \right)^2} \tag{II.61}$$

when using the visible normal sampling [Heitz and d'Eon 2014].

II.5.7 MULTILAYER BSDFS

The BSDFs introduced so far all model elementary material interaction events. To model the appearance of real materials this is not sufficient. For example an apple has a glossy reflection of a wax layer and an underlying color (yellow, red or green) which can be assumed to have Lambertian behavior. A real apple is of course even more complex.

To model the apple, the microfacet BRDF and the Lambertian BRDF could be combined. The final result of any multilayer material should be a convex combination of its contained models. For the apple example, the microfacet reflection has a share proportional to the Fresnel term F and the Lambertian part is scaled with $1 - F$. When sampling this material, either multiple rays must be traced or a random decision, often called *Russian Roulette*, must be performed. If possible, the probability of a layer should match its contribution to the final result.

Microfacet BRDF Sec. II.5.5 p. 49

Lambertian BRDF Sec. II.5.3 p. 44

Unfortunately, this is not possible for the simple example. The contribution of the reflective layer is F, which itself depends on the half-vector which is generated during the sampling of exactly this layer. That means F is not yet known at the time the decision between layers is made. It seems possible to first sample $\boldsymbol{h}$ and then decide based on $F(\boldsymbol{h})$. However, if the decision is to sample the diffuse layer instead, the final outgoing direction is chosen independently. Then, the half-vector between the new excitant and the incident direction is different from $\boldsymbol{h}$ which was used for the sampling decision in the first place. This leads to an ambiguity between PDF and a given pair of directions.

Hence, the choice of the layer is often made with a simple heuristic like a constant 50% chance for any of two layers. It is a good idea to include

the total amount of reflected light into the decision. For example a dark gray diffuse layer (say $\varrho = 0.25$) will absorb 75% of the light. If combined with a fully reflecting microfacet layer ($\varrho = 1$) the probability to choose the reflection should be $1/(1 + 0.25) = 80\%$ according to its expected relative contribution. This can still be far from the final contribution which is weighted with the Fresnel term.

There is one combination for which the decision can be made after sampling the half-vector, namely the joint microfacet BRDF with the microfacet BTDF. They apply a deterministic reflection or refraction on a facet. I.e. while the operators and with them the PDFs are different, they both keep the half-vector consistent. Therefore, (rough) transparent surfaces can be sampled optimally by including the Fresnel term for the decision between the two operators.

Microfacet BTDF Sec. II.5.6 p. 53

Figure II.19 shows examples of the described models including some multilayer combinations. The images (a) to (e) were all rendered without the Fresnel term ($F = 1$) while the combined examples all use the dielectric Fresnel function (II.44).

II.5.8 Energy Loss in Material Models

Some material models, especially microfacet-based ones, tend to lose energy, so they are not energy preserving. In microfacet models this happens because of missing multiple scattering between the facets. Technically, those models absorb a photon if it does not bounce away from the surface in the first interaction, although it might do this in reality after hitting other microscopic facets.

Microfacet models:
Oren-Nayar II.5.4 p. 44
Reflection II.5.5 p. 49
Reflection II.5.6 p. 53

Heitz et al. [2016b] developed the ground truth solution by using a random process for sampling and evaluation. It is impractical for real renderers due to its high costs. There is a much older BRDF capable of handling the energy loss from Kelemen and Szirmay-Kalos [2001]. It heavily relies on tabulated values and was not designed with respect to transmission. However, it is possible to reduce the number and size of required tables and to generalize the approach to the entire problem [Kulla and Estevez 2017]. The solution requires one table for the average albedo dependent on the roughness α and a second for the directional albedo dependent on α and $\cos\theta_\downarrow$. It then uses the two tables to define a lobe

Directional albedo Eq. (II.38) p. 40

$$\rho_{r\text{lost}}(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = \frac{(1 - \varrho(\boldsymbol{d}_\downarrow))(1 - \varrho(\boldsymbol{d}_\uparrow))}{\pi(1 - \varrho)} \tag{II.62}$$

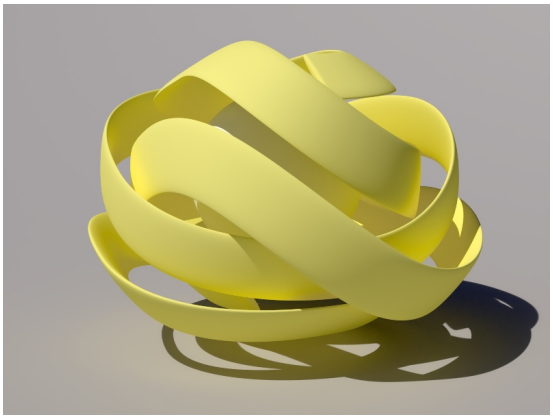
with the missing energy which is added to the microfacet BRDF or BTDF. Different tables are necessary for different models and microfacet distribution functions.

Rescaling the Oren-Nayar Model

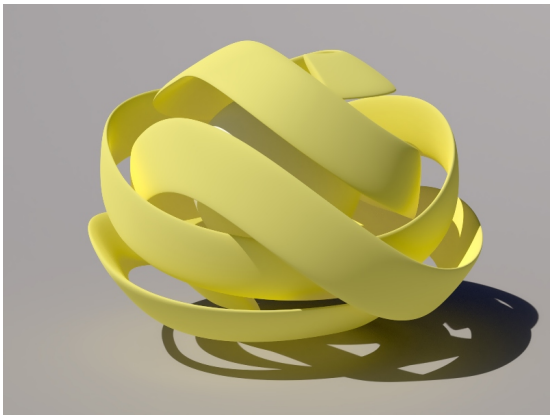
The qualitative model from Equation (II.48) loses energy and gets darker for increasing roughness values too (see Figure II.20). Using the solution of Kulla and Estevez requires a full 2D lookup table for the directional albedo $\varrho(\boldsymbol{d})$ which depends on $\cos\theta$ and the roughness α.

Directional albedo Eq. (II.38) p. 40

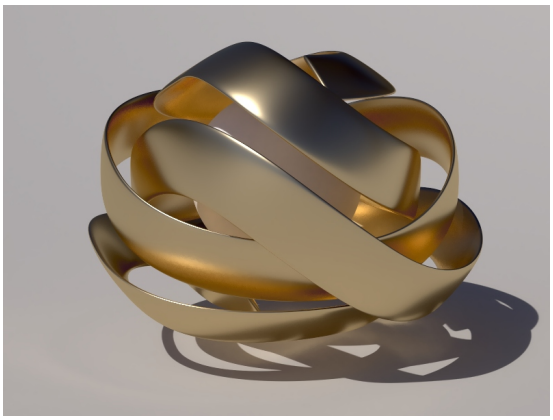
I applied an even simpler fix: the rescaling of the entire model by the average albedo ($\rho'_{\text{ON}} = \rho_{\text{ON}}/\varrho$). Thus, the model as whole becomes energy



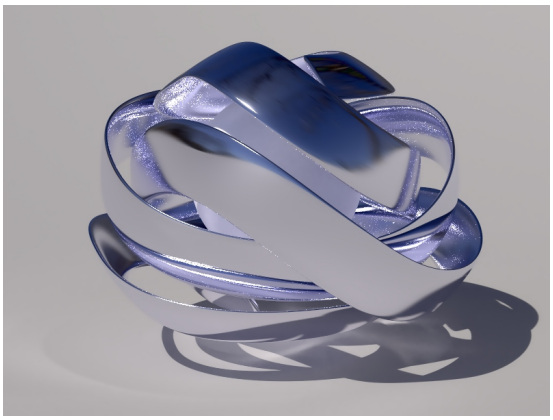
(a) Lambertian diffuse



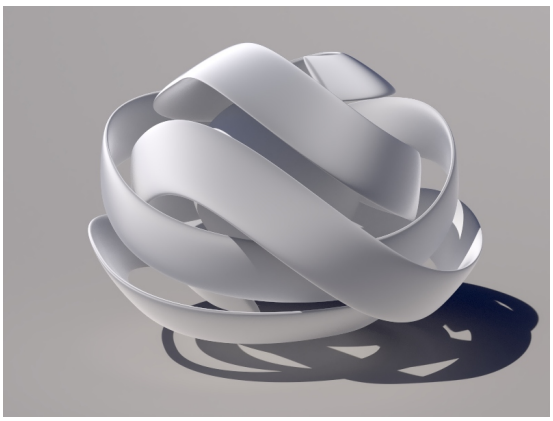
(b) Oren-Nayar diffuse



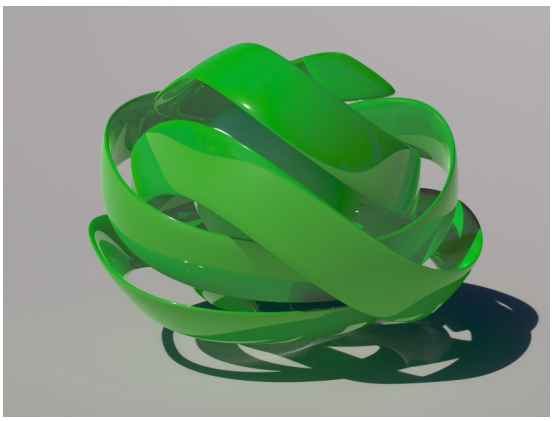
(c) Torrance-Sparrow (GGX)



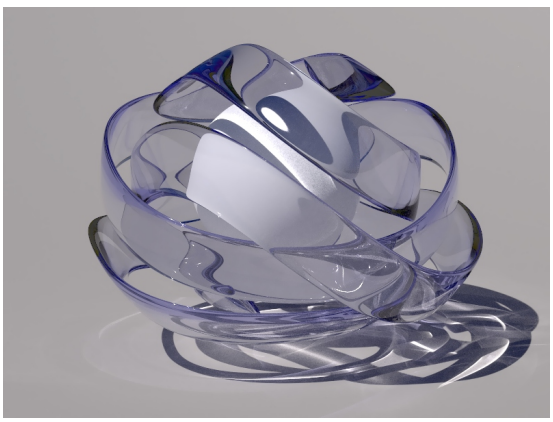
(d) Torrance-Sparrow (anisotropic BS)



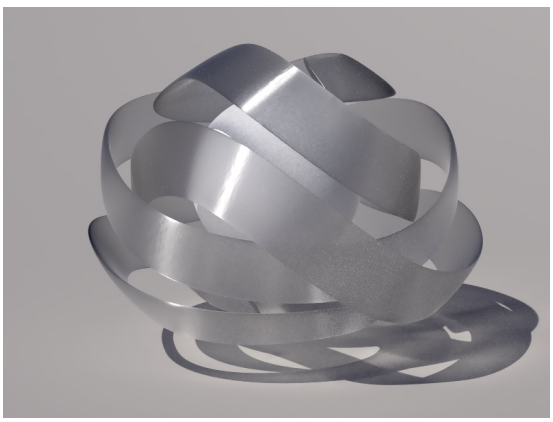
(e) Torrance-Sparrow (rough BS)



(f) Torrance-Sparrow (GGX) + Lambertian diffuse

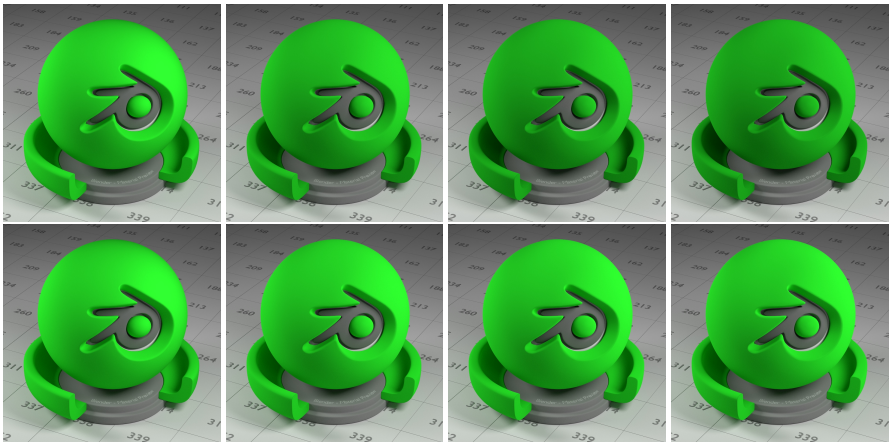


(g) Torrance-Sparrow + Walter (isotropic GGX)



(h) Torrance-Sparrow + Walter (anisotropic GGX)

Figure II.19: Example materials using the models from this section.



$\alpha = 0$ (Lambert) $\alpha = \pi/4$ $\alpha = \pi/2$ $\alpha = \pi$

Figure II.20: Energy loss of the Oren-Nayar model. The bottom row shows the results for the proposed fix – a simple rescaling of the entire model.

preserving while energy conservation is violated for certain directions. Directions for which the directional albedo is larger than the average will get a new albedo greater one. Due to reciprocity, energy from this direction will be amplified in the compensated model. Nevertheless, I found the fix working well as is shown in the bottom row of Figure II.20.

Since there is no closed form solution for the average albedo integral, I computed some values numerically. Those are given in the Appendix A.2.1.

II.5.9 FURTHER READING

Production renderer often use many more BSDFs and layers for a wider range of realistic materials. Some examples: Modern car paints often have metallic flakes inside a clearcoat layer which become a macroscopic effect in the close view [Günther et al. 2005; Rump et al. 2008]. For the rendering of scratches one even needs to model effects from wave optics as done by Werner et al. [2017]. For organic substances like hair [Chiang et al. 2016; Khungurn and Marschner 2017; Pekelis et al. 2015] or skin [d'Eon and Luebke 2007] more layers are required to capture the overall appearance.

An important effect not explained in this section is subsurface scattering. Subsurface scattering appears if light is scattered inside the medium multiple times until it leaves the object at a different point. This behavior can be observed in many real substances like milk, skin or marble. A first approximative model was introduced by Jensen et al. [2001]. It uses a dipole point source to approximate the multiple scattering between two points. Jakob et al. [2010] derive a similar model which improves the appearance of anisotropic media and can also be implemented as an (anisotropic) dipole source. Today, a common practice is to use general path tracing solutions instead of specialized models [Fong et al. 2017].

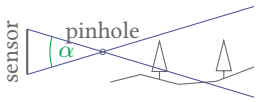
II.6 SCENE MODELING: CAMERAS

A camera is a system which projects rays in a scene onto an image plane where the image is recorded from a film or sensor. In reality a camera consists of a number of lenses through which light is refracted multiple times. This process may cause various distortions, chromatic aberrations and darkening towards the border of the lens. A good introduction to realistic models, along with further references, can be found in the PBRT book [Pharr et al. 2017, pp. 377–397]. In this thesis only the most fundamental models of the pinhole and the thin lens camera are used, since camera models are orthogonal to the introduced algorithms.

All camera models have some basic properties in common concerning their placement in the world. There is a position and an orthogonal matrix describing its orientation. Both together define the *camera space* which will be a left-hand-system in the following. The image x- and y-axis are the x/y-axis of the *camera space*, where the x-axis points to the right and the y-axis, also called *up vector*, points upwards. Finally, we have a view direction along the positive z-axis. Beyond that each model specifies its own set of parameters for the projection.

II.6.1 PINHOLE CAMERA

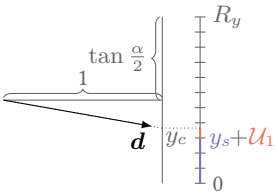
The pinhole camera performs a perspective projection where distant objects appear smaller than close ones. Therefore, all rays from the sensor pass through an infinitesimal hole. This leads to an infinite *depth of field*, meaning that all points from the camera plane to infinity are sharp. Such a camera is physically implausible, because the probability of a photon going through the hole would be zero.



As only additional parameter the pinhole camera has an opening angle α to define the *Field of View* (FOV). Since the sensor may have a rectangular size, the FOV differs for the x and y direction. In the following α is the opening angle in vertical direction.

The algorithm to sample a pinhole camera is simple. For a chosen pixel (x_s, y_s) a sub-pixel position is sampled. We can always assume a fixed pixel, because we want to solve the irradiance integral for each pixel. It is also possible to start camera rays at completely random positions on the sensor, but this would only increase the variance. Hence, we need two random numbers $\mathcal{U}_0, \mathcal{U}_1$ to find a sub-pixel which gives us a position on the sensor with a uniform PDF $p = 1$ over the pixel. Then this pixel is remapped to a direction in the *camera space*.

$$x_c = \left(2 \cdot \frac{x_s + \mathcal{U}_0}{R_x} - 1\right) \cdot \tan\frac{\alpha}{2} \cdot \frac{R_x}{R_y}$$
$$y_c = \left(2 \cdot \frac{y_s + \mathcal{U}_1}{R_y} - 1\right) \cdot \tan\frac{\alpha}{2}$$
$$\boldsymbol{d} = \frac{(x_c, y_c, 1)^T}{\sqrt{x_c^2 + y_c^2 + 1}}$$



First, the point on the sensor in $[0, R_x]\times[0, R_y]$ is remapped to the interval $[-\tan\frac{\alpha}{2}, \tan\frac{\alpha}{2}]^2$ where $R_{\{x,y\}}$ is the resolution of the sensor. For the x-coordinate it is also necessary to apply the *aspect ratio* R_x/R_y because the FOV α is specified for the vertical direction. The result is a position $(x_c, y_c, 1)^T$ on a fictive plane with a distance of one to the pinhole. After normalization we get the ray direction in camera space which can be easily transformed into *world space* using a rotation.

The resulting PDF can be found with the Jacobian method. The interval scaling results in

Domain change II.2.6 p. 28

$$|\det \boldsymbol{J}((\mathcal{U}_0, \mathcal{U}_1) \to (x_c, y_c))| = \left(\frac{R_y}{2\tan\frac{\alpha}{2}}\right)^2$$

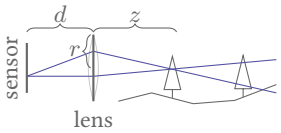
where a derivation for the scaling Jacobian can be found in Appendix A.1.1. As second transformation the normalization operator is used which is shown in Appendix A.1.3. The translations by x_s, y_s and 1 and the final rotation do not change the PDF. Thus, the final PDF is:

Scaling Jacobian A.1.1 p. 191

Normalization Jacobian A.1.3 p. 192

$$p(\boldsymbol{d}) = \frac{1}{\cos\theta_\uparrow^3}\left(\frac{R_y}{2\tan\frac{\alpha}{2}}\right)^2 = \frac{1}{\boldsymbol{d}.z^3}\left(\frac{R_y}{2\tan\frac{\alpha}{2}}\right)^2 \tag{II.63}$$

II.6.2 Thin Lens Camera



With the pinhole camera all objects appear perfectly sharp. It has an infinitely large depth of field, directly starting at the pinhole. In contrast, real imaging systems always have some amount of depth-dependent blur. The thin lens camera assumes a single perfect, planar lens which has a finite aperture. It is still a simplifying model, but it has the capability of producing depth of field effects.

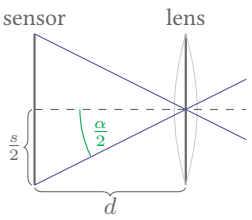
The lens is described by two parameters: the focal length f and the lens radius r. The greater the focal length, the smaller the FOV. The parameter r determines the amount of blur, where larger radii cause a larger *circle of confusion*. Additionally, we need a parameter z for the focus distance. All points on the plane at this distance will be perfectly sharp, while closer or more distant points will be blurred stronger. Dependent on the desired focus distance, the distance d between sensor and lens must be adjusted. The two distances are connected by the *Gaussian lens equation*

$$\frac{1}{d} - \frac{1}{z} = \frac{1}{f} \tag{II.64}$$

which gives

$$d = \frac{fz}{f+z} \tag{II.65}$$

for the required lens to sensor distance.



Using basic trigonometry it is also possible to compute the field of view α, which depends on the parameters f, z and the vertical sensor size s:

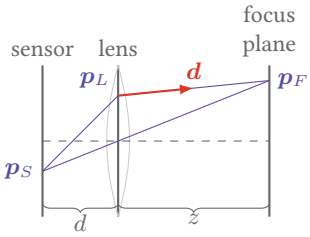
$$\alpha = 2\arctan\frac{s}{2d} = 2\arctan\frac{s\cdot(f+z)}{2fz}.$$

To sample the thin lens camera, we need to sample two positions – one on the sensor (pixel, $\boldsymbol{p}_S$) and one on the lens ($\boldsymbol{p}_L = (x_L, y_L, 0)^T$). The position on the lens defines the origin of the ray, whereas the direction must be computed from both positions. First, we sample a sub-pixel position $\boldsymbol{p}_S = (x_S, y_S, z_S)^T$ on the sensor in camera space, similar to the pinhole camera

$$\begin{aligned} x_S &= \left(2 \cdot \frac{x_s + \mathcal{U}_0}{R_x} - 1\right) \cdot \frac{s}{2} \cdot \frac{R_x}{R_y} \\ y_S &= \left(2 \cdot \frac{y_s + \mathcal{U}_1}{R_y} - 1\right) \cdot \frac{s}{2} \\ z_S &= -d \end{aligned}$$

Chosen pixel (x_s, y_s)
Resolution
$R_x \times R_y (quadratic pixels)$

Now, we need the intersection point $\boldsymbol{p}_F$ between the focus plane and a ray through the lens center. Both the direction to be generated and the central ray must intersect in this point to produce a sharp image. This condition gives $\boldsymbol{p}_F = -\boldsymbol{p}_s \cdot z/d$ which leads to final direction vector



$$\begin{aligned} \boldsymbol{d} = \frac{\boldsymbol{p}_F - \boldsymbol{p}_L}{\|\boldsymbol{p}_F - \boldsymbol{p}_L\|} &= \frac{(-x_S \cdot \frac{z}{d} - x_L, y_S \cdot \frac{z}{d} - y_L, z)^T}{\left\|(-x_S \cdot \frac{z}{d} - x_L, y_S \cdot \frac{z}{d} - y_L, z)^T\right\|} \\ &= \frac{(-x_S - x_L \cdot \frac{d}{z}, y_S - y_L \cdot \frac{d}{z}, d)^T}{\left\|(-x_S - x_L \cdot \frac{d}{z}, y_S - y_L \cdot \frac{d}{z}, d)^T\right\|}. \end{aligned} \tag{II.66}$$

The scaling in the second line is only to simplify the following derivation of the PDF.

To find the PDF, we successively apply the Jacobian method again. The sampling of the position $\boldsymbol{p}_S$ is a scale

$$|\det \boldsymbol{J}((\mathcal{U}_0, \mathcal{U}_1) \to (x_S, y_S))| = \left(\frac{R_y}{s}\right)^2$$

analogously to that in the pinhole model. The transformation into a direction as defined in Equation (II.66) consists of an constant offset ($-\boldsymbol{p}_L \cdot d/z$) which does not change the PDF and the normalization with the Jacobian from Appendix A.1.3. The product of the two gives

$$p(\boldsymbol{d}) = \frac{d^2}{\cos\theta_\uparrow^3}\left(\frac{R_y}{s}\right)^2 = \frac{d^2}{\boldsymbol{d}.z^3}\left(\frac{R_y}{s}\right)^2 \tag{II.67}$$

which depends on the sensor size s and the sensor to lens distance d. Note that the angle $\theta_\uparrow$ is that between the direction from lens to object ($\boldsymbol{d}$) and the central view direction ($(0,0,1)^T$ in local coordinates).

II.7 SURFACE LIGHT TRANSPORT

Section II.1 introduced the general formulas to describe light in space and with respect to surfaces. In Sections II.3 to II.5 models for light sources and objects were introduced and most importantly the concept of BSDFs ρ. Bringing both together, the light transport problem can be expressed as a single equation.

II.7.1 THE RENDERING EQUATION

The target of light transport simulation is to determine the excitant *radiance* $L(\boldsymbol{x}, \boldsymbol{d}_\uparrow)$ of an observed surface point $\boldsymbol{x}$. We know how to express the illumination as differential *irradiance* and that the BSDF transforms differential *irradiance* into excitant differential *radiance*. By integrating over this product we get the total excitant *radiance*

Radiance Eq. (II.8) p. 17

Differential irradiance from Eq. (II.5) p. 16
BSDF ρ defined Eq. (II.34) p. 39

$$L(\boldsymbol{x}, \boldsymbol{d}_\uparrow) = \int_\Omega \rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)\, \mathrm{d}E(\boldsymbol{x}, \boldsymbol{d}_\downarrow). \tag{II.68}$$

This is not yet the common form of the rendering equation. By taking (II.9) we can also express the equation in terms of incident *radiance*

$E(\boldsymbol{x}) \leftrightarrow L(x, \boldsymbol{d}_\downarrow)$ from Eq. (II.9) p. 17

$$L(\boldsymbol{x}, \boldsymbol{d}_\uparrow) = \int_\Omega \rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) L_\downarrow(\boldsymbol{x}, \boldsymbol{d}_\downarrow)\, |\cos\theta_\downarrow|\, \mathrm{d}\omega.$$

Due to the invariance of *radiance* over the distance we can further exchange $L_\downarrow(\boldsymbol{x}, \boldsymbol{d}_\downarrow)$ with the excitant *radiance* $L(\boldsymbol{x}', -\boldsymbol{d}_\downarrow)$ of the point $\boldsymbol{x}'$ observed in the incident direction. Finally, the surface can add radiation from different processes like black body radiation which we account for by adding a self-radiation term $L_e(\boldsymbol{x}, \boldsymbol{d}_\uparrow)$. Thus, the final form of the rendering equation is

$$L(\boldsymbol{x}, \boldsymbol{d}_\uparrow) = L_e(\boldsymbol{x}, \boldsymbol{d}_\uparrow) + \int_\Omega \rho(\boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) L(\boldsymbol{x}', -\boldsymbol{d}_\downarrow)\, |\cos\theta_\downarrow|\, \mathrm{d}\omega. \tag{II.69}$$

and was used the first time by Kajiya [1986] in an equivalent, three-point form (observer, surface $\boldsymbol{x}$, sender $\boldsymbol{x}'$).

II.7.2 PATH MEASUREMENT FORMULATION

The rendering equation describes the light transport recursively, which can be resolved into a formulation with sums [Veach 1997, chapter 8.2]. Many problems can be described more easily in the resulting path-space.

Each point of interaction is called a vertex and is associated with a position $\boldsymbol{x}$ and further properties (for example a BSDF $\rho(\boldsymbol{x}, \dots)$). Following a single differential direction in each recursion forms a *path* $\mathcal{P}$ which is a sequence of vertices $\boldsymbol{x}_0 \boldsymbol{x}_1 \dots \boldsymbol{x}_\ell$.

Let $\mathcal{P}^\ell$ be a path of ℓ segments with $\left|\mathcal{P}^\ell\right| = \ell + 1$ vertices. The self-radiation L_e can be seen as the result of a path $\mathcal{P}^0$. Paths $\mathcal{P}^1$, ending on a light source, describe the direct illumination.

Then, the path-space is the union of all finite paths

$$\bar{\mathcal{X}} = \bigcup_{\ell=0}^{\infty} \{\mathcal{P}_i^\ell : \forall i\}. \tag{II.70}$$

From those we are interested in the subset $\mathcal{X} \subset \bar{\mathcal{X}}$ of all path $\mathcal{X} = \{\mathcal{P} : \mathcal{P} \in \bar{\mathcal{X}}, \boldsymbol{x}_0(\mathcal{P}) = \boldsymbol{x}\}$ beginning in the same point $\boldsymbol{x}$. Note that I use $\boldsymbol{x}_0$ for the camera point and not as the light emitting point, which is also often done in literature including Veach's thesis. Respectively, the light emitting point becomes $\boldsymbol{x}_\ell$ in my notation.

Veach [1997, chapter 8.2] derived an integral formulation on the path-space by expanding the rendering equation. It reads

Rendering Equation from Eq. (II.69) p. 63
Estimator I from II.2 p. 19

$$I = \int_{\mathcal{X}} f(\mathcal{P}) \, \mathrm{d}\mu(\mathcal{P}) \tag{II.71}$$

where f is the path *measurement contribution function* and

$$\mathrm{d}\mu(\mathcal{P}^\ell) = \mathrm{d}A(\boldsymbol{x}_0) \dots \mathrm{d}A(\boldsymbol{x}_\ell)$$

$\mathrm{d}A(\boldsymbol{x})$ is the differential area around point $\boldsymbol{x}$

is the *area-product measure*. The *measurement contribution function* itself is

$$\begin{aligned} f(\mathcal{P}^\ell) = & W(\boldsymbol{x}_0, \boldsymbol{d}_0^{\rightarrow}) \cdot G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \\ & \cdot \prod_{i=1}^{\ell-1} \rho(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow}) \cdot G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) \\ & \cdot L_e(\boldsymbol{x}_\ell, \boldsymbol{d}_\ell^{\leftarrow}) \end{aligned} \tag{II.72}$$

$$\boldsymbol{d}_i^{\rightarrow} = \frac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_i}{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|}$$

$$\boldsymbol{d}_i^{\leftarrow} = \frac{\boldsymbol{x}_{i-1} - \boldsymbol{x}_i}{\|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|}$$

$$\cos\theta^{\{\rightarrow,\leftarrow\}} = \left\langle \boldsymbol{n}, \boldsymbol{d}^{\{\rightarrow,\leftarrow\}} \right\rangle$$

where

$$G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) = \frac{\left|\cos\theta_i^{\rightarrow} \cdot \cos\theta_{i+1}^{\leftarrow}\right|}{\|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|^2} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \tag{II.73}$$

is the transport factor of the segment. Most of the terms are obtained by the photometric distance law and the surface interaction (ρ and cosines). W is a new function which weights the sensor response at the point of interest in the respective direction. For example, the pinhole camera has $W = p$, because the throughput of sampling at the camera W/p is usually set to one.

Photometric distance law from Eq. (II.5) p. 16

Pinhole camera II.6.1 p. 59

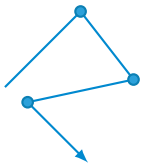
The term V is also a new function according for the mutual visibility between two points. If there is any opaque surface on the segment between the two points, V is 0 and the path does not contribute any light. This is also known as shadowing. While this thesis focuses on surface transport it is easy to model an absorbing medium between the two points by using any $V \in [0, 1]$. This allows to describe substances like clean gases or water, which do not show scattering, but still absorb part of the light.

To simplify many of the formulas building on the path formulation I want to introduce the function

$$\wp(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow}) = \begin{cases} W(\boldsymbol{x}_0, \boldsymbol{d}_0^{\rightarrow}) & \text{if } i = 0 \\ L(\boldsymbol{x}_\ell, \boldsymbol{d}_\ell^{\leftarrow}) & \text{if } i = \ell \\ \rho(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow}) & \text{else} \end{cases} \tag{II.74}$$

which unifies the writing of W, L_e and ρ. Also, it is necessary to define the incident directions on the start vertices $\boldsymbol{d}_0^{\leftarrow} = \boldsymbol{d}_\ell^{\rightarrow} = \boldsymbol{0}$ for the unified usage of $\wp$. These are never used in any computation (due to the definition of $\wp$).

II.7.3 RANDOM WALKS



In the next step we want to solve the *rendering equation* (II.69) by using Monte Carlo integration. Starting either at the sensor or the emitter, we can locally sample excitant directions with the methods and density functions given in the previous sections. Then, a ray in this direction is traced to find the next vertex on the path. This common operation is called *random walk*.

If no next point is found the walk terminates. Also, a walk can be explicitly terminated after a maximum number of steps to avoid infinite computation times. This introduces a bias due to the lost energy of longer paths.

Additionally, a method to increase the effectiveness of a sampler is to randomly terminate paths via Russian roulette. If a path has a small throughput, it will likely have a small contribution. Investing the same computation time into other paths will reduce the variance faster, so we discard low throughput samples with a higher probability. If a sample is discarded with probability P_{RR} its throughput must be divided by the very same P_{RR}. Doing so, Russian roulette is an unbiased path termination criterion. The surviving samples will carry all the lost energy from the discarded ones. The most sophisticated choice for P_{RR}, known to me, is the *Adjoint-driven Russian Roulette* [Vorba and Křivánek 2016].

Def. Path Russian roulette Throughput Sec. II.2.2 p. 20

If a *random walk* starts at the sensor it will be called *view sub-path*. Else, if it starts at the emitter it will be called *light sub-path*. W.l.o.g. let us consider the *view sub-paths* and hits on light sources only. Every observation which can be made for one path type also applies to the other one.

Once the paths extends to an emitter $L_e > 0$, a full path is established. The contribution of such a sample is

$$t_{\text{rv}} = \frac{W(\boldsymbol{x}_0, \boldsymbol{d}_0^{\rightarrow})\,|\cos\theta_0^{\rightarrow}|}{p(\boldsymbol{x}_0)p(\boldsymbol{d}_0^{\rightarrow})} \cdot V_0 \cdot \left[\prod_{i=1}^{\ell-1} \frac{\rho(\boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow})\,|\cos\theta_i^{\rightarrow}|}{p(\boldsymbol{d}_i^{\rightarrow})} \cdot V_i\right] \cdot L_e(\boldsymbol{x}_\ell, \boldsymbol{d}_\ell^{\leftarrow})$$

Def. Random hit throughput (view path)

$$= \frac{1}{p(\boldsymbol{x}_0)} \cdot \left[\prod_{i=0}^{\ell-1} \frac{\wp(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow})\,|\cos\theta_i^{\rightarrow}|}{p(\boldsymbol{d}_i^{\rightarrow})} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})\right] \cdot L_e(\boldsymbol{x}_\ell, \boldsymbol{d}_\ell^{\leftarrow}) \tag{II.75}$$

Applying unified notation $\wp$ defined in Eq. (II.74) p. 64

by using the fundamental f/p terms of the individual Monte Carlo events in the *random walk*. The visibility terms are included to account for volumetric attenuation only. There is no shadowing, because each sampling event will always hit the next visible vertex – wherever this is. In vacuum these terms can be omitted. The probability $p(\boldsymbol{x}_0)$ is the PDF of sampling the start position. It is one for pinhole cameras and a per area density for finite lens cameras.

For a light sub-path contribution all the sampling probabilities are reversed and the result in the simplified notation is

$$t_{\text{rl}} = W(\boldsymbol{x}_0, \boldsymbol{d}_0^{\rightarrow}) \cdot \left[\prod_{i=1}^{\ell} \frac{\wp(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow})\,|\cos\theta_i^{\leftarrow}|}{p(\boldsymbol{d}_i^{\leftarrow})} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i-1})\right] \cdot \frac{1}{p(\boldsymbol{x}_\ell)}. \tag{II.76}$$

Def. Random hit throughput (light path)

Analogously to $p(\boldsymbol{x}_0)$, $p(\boldsymbol{x}_\ell)$ is the probability density to sample the first vertex of the light sub-path. It is a discrete probability for point lights and a per area density for area lights.

This kind of contribution will be called *random hit* in the following. The sub-indices rv and rl of t denote ***r**andom hit **v**iew sub-path* and ***r**andom hit **l**ight sub-path*.

The throughput values t are the actual values of individual samples of the rendering process which are often computed directly. However, it is also useful to derive the path-space PDF of these samples. We obtain it by inverting the definition of the throughput $p = f/t$ where f is the measurement contribution function. After shortening all redundant terms like the visibility we get

Measurement function
Eq. (II.71) p. 64

$$p_{\mathrm{rv}}(\mathcal{P}^\ell) = p(\boldsymbol{x}_0) \cdot \prod_{i=0}^{\ell-1} p(\boldsymbol{d}_i^{\rightarrow}) \cdot \frac{\left|\cos\theta_{i+1}^{\leftarrow}\right|}{\left\|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\right\|^2}$$
$$= p(\boldsymbol{x}_0) \cdot \prod_{i=0}^{\ell-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i) \tag{II.77}$$

$$p_{\mathrm{rl}}(\mathcal{P}^\ell) = p(\boldsymbol{x}_\ell) \cdot \prod_{i=1}^{\ell} p(\boldsymbol{d}_i^{\rightarrow}) \cdot \frac{\left|\cos\theta_{i-1}^{\rightarrow}\right|}{\left\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\right\|^2}$$
$$= p(\boldsymbol{x}_\ell) \cdot \prod_{i=1}^{\ell} p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i) \tag{II.78}$$

for the two above sampling methods. The term $\cos\theta/\|\square\|^2$ is the Jacobian of the transport operator. It transforms the per steradian PDF $p(\boldsymbol{d})$ into a per area PDF $p(\boldsymbol{x}_\square|\boldsymbol{x})$ at the next vertex of the path. The conditional notation will be used in the following without repeating the Jacobian term.

PDF transformation
$p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i) = \frac{p(\boldsymbol{d}_i^{\rightarrow})\left|\cos\theta_{i+1}^{\leftarrow}\right|}{\left\|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\right\|^2}$
$p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i) = \frac{p(\boldsymbol{d}_i^{\leftarrow})\left|\cos\theta_{i-1}^{\rightarrow}\right|}{\left\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\right\|^2}$

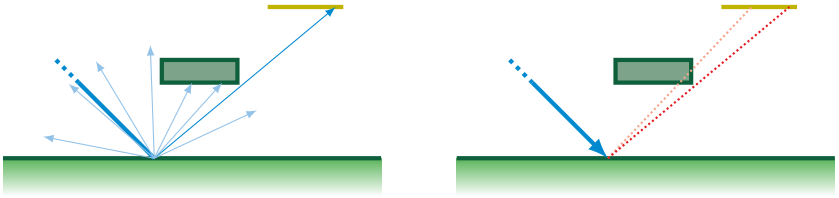
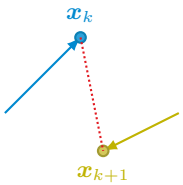


Figure II.21: Finding small emitters by random walks is unlikely and produces a high variance (left). Next event estimation (right) can be much more effective in this cases.

II.7.4 CONNECTIONS

A pure random walk algorithm gives poor results for general scenes. W.l.o.g. let us consider the *view sub-paths* as visualized in Figure II.21. A lot of computation time is wasted, if the probability to hit the emitter is small. In many situations a better method is to sample a vertex on a light source and to compute the light transport for this segment explicitly. This look-ahead computation of the random walk is called *Next Event Estimation* (NEE).



To compute the contribution of the paths, the photometric distance law can be applied almost directly. However, it is important to perform a shadow-test by tracing another ray between the point on the light source and the last vertex of the walk. The contribution of the connection between any two sub-paths is

Photometric distance law from Eq. (II.5) p. 16

$$
\begin{aligned}
t_\mathrm{c} = \frac{1}{p(\boldsymbol{x}_0)} \cdot \prod_{i=0}^{k-1} \frac{\wp_i \left|\cos\theta_i^{\rightarrow}\right|}{p(\boldsymbol{d}_i^{\rightarrow})} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \\
\cdot \frac{\wp_k \cdot \left|\cos\theta_k^{\rightarrow}\right| \cdot V(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) \cdot \left|\cos\theta_{k+1}^{\leftarrow}\right| \cdot \wp_{k+1}}{\left\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\right\|^2} \\
\cdot \left[\prod_{i=k+2}^{\ell} \frac{\wp_i \left|\cos\theta_i^{\leftarrow}\right|}{p(\boldsymbol{d}_i^{\leftarrow})} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i-1})\right] \cdot \frac{1}{p(\boldsymbol{x}_\ell)},
\end{aligned}
\tag{II.79}
$$

$\wp_i = \wp(\boldsymbol{x}_i, \boldsymbol{d}_i^{\rightarrow}, \boldsymbol{d}_i^{\leftarrow})$
V visibility Eq. (II.73) p. 64

where k is the smaller index of the two vertices in the connection. If $k = 0$ the view sub-path has only one vertex and the left product vanishes. The same happens for $k = \ell$ to the second product term. Like for the random hit event, there are the additional PDFs $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{x}_\ell)$ for the sampling of the start vertices.

For finite light sources and non-specular paths both *random walks* and *random hits* can solve the rendering problem. They have a different degree of variance and can complement each other. While *random hits* can trace specular paths, NEE is the only of the two possibilities to compute the contribution of non-physical light sources like point or directional lights.

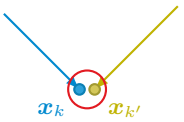
Point lights II.3.1 p. 29
Directional lights II.3.3 p. 32

Even better results are possible by combining both contribution possibilities in an optimal way. To achieve that we need another tool: multiple importance sampling. This leads to the *Path Tracing* algorithm which is described in Section II.8.3. For the MIS computation we need the sample probability, which is again derived by dividing the measurement contribution function by the throughput value t_c resulting in

MIS II.2.3 p. 22

$$
p_\mathrm{c} = p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{k-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot \left[\prod_{i=k+2}^{\ell} p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i)\right] \cdot p(\boldsymbol{x}_\ell). \tag{II.80}
$$

II.7.5 MERGES



A third possibility to create full transport paths is to *merge* the two end points of a light and a view sub-path under the assumption that both parts ended in the same point. A merge is biased, because it artificially blurs the contribution of the sub-paths, if the points are not identical. More precisely, there are multiple types of bias:

Proximity The irradiance is filtered over a finite neighborhood. This artifact is most severe at discontinuities like shadow and caustic borders.

Boundary Underestimation near borders due to missing irradiance samples in free space.

Topological Invalid irradiance from non-planar topology. Examples are light bleeding through walls and overestimations on curved surfaces.

The smaller the allowed radius for a merge, the smaller the bias regardless of its type.

The contribution of a *merge* is similar to that of a connection. It consists of the throughput from the two sampled sub-paths and the central term for the merge

$$t_\mathrm{m} = \frac{1}{p(\boldsymbol{x}_0)} \cdot \left[\prod_{i=0}^{k-1} \frac{\wp_i \left|\cos\theta_i^{\rightarrow}\right|}{p_i^{\rightarrow}} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})\right] \cdot \wp_k \cdot K(\boldsymbol{x}_k, \boldsymbol{x}_{k'}) \cdot \left[\prod_{i=k'+1}^{\ell} \frac{\wp_i \left|\cos\theta_i^{\leftarrow}\right|}{p_i^{\leftarrow}} \cdot V(\boldsymbol{x}_i, \boldsymbol{x}_{i-1})\right] \cdot \frac{1}{p(\boldsymbol{x}_\ell)}. \tag{II.81}$$

This kind of path differs from the others in that it has $\ell + 2$ vertices, since the two sub-path end-points $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k'}$ are assumed to be the same vertex.

The function K is the kernel which weights the contribution over the distance of the two points. In the following, the uniform kernel

$$K(\boldsymbol{x}_k, \boldsymbol{x}_{k'}) = \begin{cases} \frac{1}{\pi r^2} & \text{if } \|\boldsymbol{x}_k - \boldsymbol{x}_{k'}\| \le r \\ 0 & \text{otherwise} \end{cases}, \tag{II.82}$$

is used, because MIS computations for non-uniform kernels are less precise.

Combining *random hits*, *connections* and *merges* is also possible. The PDF used to achieve the combination is

MIS for merges Sec. II.8.8 p. 79

$$\begin{aligned} p_\mathrm{m} &= p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{k-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot \frac{1}{K(\boldsymbol{x}_k, \boldsymbol{x}_{k'})} \cdot \left[\prod_{i=k'+1}^{\ell} p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i)\right] \cdot p(\boldsymbol{x}_\ell) \\ &= p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{k-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot \pi r^2 \cdot \left[\prod_{i=k'+1}^{\ell} p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i)\right] \cdot p(\boldsymbol{x}_\ell). \end{aligned} \tag{II.83}$$

The reason why the kernel K must be constant is that the evaluation of p_m must result in the same value at other vertices $j \neq k$. Since the distance $\|\boldsymbol{x}_k - \boldsymbol{x}_{k'}\|$ of an assumed merge is unknown at those other vertices, it cannot be used in the evaluation of K. It is still possible to use arbitrary kernels for the renderer throughput t_m when using the constant assumption for the MIS only. However, in this case the MIS will diverge from the optimal weighting for the real sampler.

II.7.6 MIS WEIGHTS FROM SAMPLE VALUES

On of the greatest insights I had during the work on this thesis was that we can compute the same weights as in the balance or power heuristic by using the sample values t themselves instead of the probabilities p. Specific forms of sample values t were given in the previous sections on light transport operators. The heuristic using the sample values reads

Balance h. Eq. (II.23) p. 23
Power h. Eq. (II.24) p. 23
Light transport operators: *hit* Sec. II.7.3 p. 65, *connection* Sec. II.7.4 p. 67, *merges* Sec. II.7.5 p. 68

$$w_i^\beta(x) = \frac{\left(N_i/t_i(x)\right)^\beta}{\sum_{k=1}^{S}\left(N_k/t_k(x)\right)^\beta}. \tag{II.84}$$

This means that any change to the renderer which influences the sample value t must be part of the MIS. Examples are the shading normal correction or Russian roulette which are often neglected in MIS descriptions.

By inserting $t = f/p$, where f is the measurement contribution function and t, p are the sample value and sample probability, it can be shown that Equation (II.84) and Equation (II.24) are equivalent, if f is constant over all used sampling methods.

f defined in Eq. (II.72) p. 64

I published the above insight including the proof in the technical report *Path Throughput Importance Weights* [Jendersie 2018]. However, I want to present a different, more constructive proof here. It is effectively a copy of the derivation of the balance heuristic made by Veach [1997, p. 288].

We start with the multi-sample model and let

Multi-sample model Eq. (II.22) p. 23

$$F_{ij} = \frac{w_i(x_{ij})f(x_{ij})}{p_i(x_{ij})}$$

i index of sampler
j index of a sample

to simplify the notation. The expected values

$$\mu_i = \mathrm{E}[F_{ij}] = \int_{\mathcal{X}} w_i(\mathcal{P})f(\mathcal{P})\,\mathrm{d}\mu(\mathcal{P})$$

are the searched values of the integral. They are equal for all samplers i, if those are unbiased or equally biased. Then the variance of the sampler is

$$\begin{aligned}
\mathrm{V}\left[\hat{I}_{\text{MIS2}}\right] &= \mathrm{V}\left[\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}F_{ij}\right] = \sum_{i=1}^{S}\frac{1}{N_i^2}\sum_{j=1}^{N_i}\mathrm{V}[F_{ij}] \\
&= \left(\sum_{i=1}^{S}\frac{1}{N_i^2}\sum_{j=1}^{N_i}\mathrm{E}\left[F_{ij}^2\right]\right) - \left(\sum_{i=1}^{S}\frac{1}{N_i^2}\sum_{j=1}^{N_i}\mathrm{E}[F_{ij}]^2\right) \\
&= \left(\sum_{i=1}^{S}\frac{1}{N_i^2}\sum_{j=1}^{N_i}\int_{\mathcal{X}}\frac{w_i^2(\mathcal{P})f^2(\mathcal{P})}{p_i^2(\mathcal{P})}p_i(\mathcal{P})\,\mathrm{d}\mu(\mathcal{P})\right) - \left(\sum_{i=1}^{S}\frac{1}{N_i^2}N_i\mu_i^2\right) \\
&= \left(\int_{\mathcal{X}}\sum_{i=1}^{S}\frac{w_i^2(\mathcal{P})f^2(\mathcal{P})}{N_i\,p_i(\mathcal{P})}\,\mathrm{d}\mu(\mathcal{P})\right) - \left(\sum_{i=1}^{S}\frac{1}{N_i}\mu_i^2\right)
\end{aligned} \tag{II.85}$$

At this point Veach minimized the first term alone and computed bounds for the different heuristics on the error made by not incorporating the right term. Further, he assumed that f is independent of the sampler such that it can be removed in the optimization. I proceed in the same way but replace the quotient f/p_i with t_i.

To minimize the integral on the left it suffices to minimize the sum independent of the sample path $\mathcal{P}$. Thus, we omit the argument $\mathcal{P}$ and minimize

$$\sum_{i=1}^{S} \frac{w_i^2 \, f \, t_i}{N_i}$$

applying the Lagrange multiplier method with respect to $\sum w_i = 1$. Thus, the Lagrange function is

$$\mathcal{L} = \sum_{i=1}^{S} \frac{w_i^2 \, f \, t_i}{N_i} - \lambda \left(1 - \sum_{i=1}^{S} w_i \right).$$

To find the local minimum we take the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{2 w_i f t_i}{N_i} + \lambda$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = -1 + \sum_{i=1}^{S} w_i$$

which must be set zero, ${}^{\partial \mathcal{L}}/_{\partial w_i} = \mathbf{0}$ and ${}^{\partial \mathcal{L}}/_{\partial \lambda} = 0$, to obtain the system of linear equations

$$\forall i : w_i + \frac{N_i}{2 f t_i} \lambda = 0$$
$$\sum w_i = 1.$$

Eliminating all w_i in the last equation by subtracting all others yields

$$-\lambda \sum \frac{N_i}{2 f t_i} = 1$$
$$\lambda = -\frac{1}{\sum {}^{N_i}/_{2 f t_i}}$$

which can be inserted back to the other equations. We obtain the result

$$w_i - \frac{{}^{N_i}/_{2 f t_i}}{\sum {}^{N_k}/_{2 f t_k}} = 0$$
$$w_i = \frac{{}^{N_i}/_{t_i}}{\sum {}^{N_k}/_{t_k}}$$

Hence, the only difference between the known form of the balance heuristic and the one using the estimators is an early removal of the constant path measurement contribution function f.

II.8 Light Transport Methods

Using MIS we can combine the three basic operations *random walk*, NEE and *merge*. This section systematically introduces the rendering methods obtained by the different combinations.

To describe the paths which can be found by the respective method, Heckbert's notation [Heckbert 1990] is commonly used. It forms regular expressions of the symbols L, E, S and D, where L and E mark the light source and sensor (eye) end points of the path.

In his thesis [1997, p. 231], Veach formalized the meaning of D (diffuse) and S (specular/deterministic). A specular vertex has a singularity which means that its distribution is a Dirac delta. All other vertices count as diffuse, although they might behave like specular vertices in practical applications. In theory, a sampler which ends its random walk on any finite interaction vertex is possible and unbiased. Only evaluations and random hits of infinitesimal interactions (specular) cause a sampler to totally miss certain light effects. In practice however, a small finite event can cause arbitrarily high variance which makes the results unusable. Thus, the paths which a sampler can find with a sufficient quality differ from the paths it can find given an infinite amount of computation time.

Dirac delta II.3.3 p. 32

We often call the more difficult diffuse vertices *glossy*. However, there is no clear distinction whether a vertex should count as diffuse or glossy. Informally, a glossy vertex has a distinctive peak in its distribution. This peak can cause unpractical high variance, a problem which is further discussed in Chapter V along with a solution to reduce the variance of glossy events. Within this chapter I will use the symbol G to mark specific locations in the regular expressions when discussing specific samplers.

Veach also extended Heckbert's notation to describe light sources more precisely. Thereby, each end point is described by three symbols: L or E to mark the type and any combination of the two symbols S and D for the position and the outgoing direction. The following table gives the interpretation of specific symbol combinations.

LDD	**Area light**: finite extent, Lambertian directional emission **Environment light**: infinite extent, all directions (non-uniform)
LSD	**Point light**: deterministic position, uniform emission
LDS	**Parallel light**: infinite extent, deterministic direction
LSS	**Laser beam**: deterministic position and direction
DDE	**Realistic camera**: directions in frustum, finite sensor extent
DSE	**Pinhole camera**: directions in frustum, deterministic position
SDE	**Orthographic camera**: deterministic direction, finite extent

The end vertices are distinguished from the other vertices through the coloring. If any end vertex is possible, L is used as short form for L(S|D)^2 and E for (S|D)^{2}E. For example the set of all possible paths is noted as L(S|D)*E. This *full-path regular expression* is used in this chapter.

$\square^*$: zero or more
$\square^+$: one or more
$\square^n$: exactly n
| : or (alternative)
(·) : (S|D)

From the set of all possible paths L(·)*E, there are three subsets with specific names: *caustics*, *SDS* and *specular* paths. All three are challenging for many of the sampling approaches and often cause most of the variance problems in a rendering.

Caustics are directly visible spots of multiple specularly reflected light. They are described by LS^+DE.

SDS paths are caustics with at least one specular reflection on sensor side. They are named after their regular expression LS^+DS^+E. However, it makes sense to include even longer chains which have at most one diffuse vertex in a row $L(S^+D)^+S^+E$, because these paths have the same sampling complexity.

Specular paths have no diffuse intermediate vertices (LS^+E) and are the most difficult case. They have a non-zero contribution, but are not sample-able because the chance to find a specular path is always zero. For the other two cases the set of samplers defines if the path is sample-able in theory and practice.

Using the auxiliary symbol G, an area light might be LDD or LGD, mostly depending on its size. Clearly, it is simpler to randomly hit large light sources than small ones. In the limit a small area light LGD becomes a point light LSD. With respect to practical applications both LGD and LSD lights cause the same kind of problems. Long before reaching the limit, the variance increases beyond feasible values. In the same way we can define *glossy paths* LG^+E, *glossy caustics* LG^+DE and GDG paths $L(G^+D)^+G^+E$ which behave equally bad as their specular counter parts with respect to practical methods. Details depend on the specific algorithm and are discussed in the following sub-sections.

II.8.1 UNIDIRECTIONAL PATH TRACING

As mentioned in Section II.7.3 tracing only paths from the sensor until randomly hitting emitters already gives a first transport algorithm: *unidirectional Path Tracing* (uPT). It computes an unbiased and consistent solution, if the random numbers used for the successive sampling events are independent. Using a forced termination of *random walks* introduces a bias which is the missing contribution of paths longer than the used maximum.

uPT is able to find paths of the form LDD(·)*E. Those paths have a finite, and therefore hitable, light source. Even in theory, uPT will not find LSD, LDS or LSS light sources, leading to a biased estimator for certain scenes. In practice, uPT will also fail for scenes with small area lights LGD or very focused environment maps LDG. In the limit, area lights become point lights and environment lights become parallel lights. Figure II.22 shows an example where the increasing variance for a shrinking area light is observable.

II.8.2 UNIDIRECTIONAL LIGHT TRACING

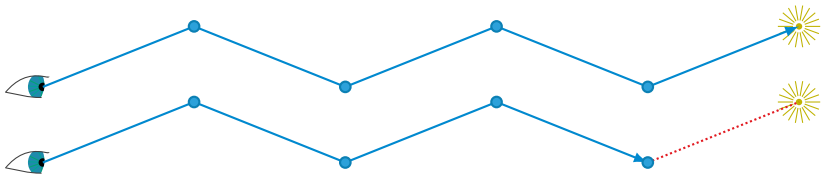
The second mentioned method is the *unidirectional Light Tracing* (uLT) which reverts the direction of the *random walk*. Opposed to uPT, it finds paths regardless of the lights source type, but only for hitable cameras: L(·)*DDE. It is useless in presence of the non-physical pinhole camera which cannot be hit randomly. For sensors with a real, but small extent this method tends to have a high variance. Therefore, the raw algorithm is never applied, but could appear in connection with other methods.

One of the first methods to perform *random walks* beginning at light sources is the backward ray tracing from [Arvo 1986]. It is a two pass method which stores the contribution of light paths into a map on the surface. In the second pass forward ray tracing (beginning at the sensor) is performed, whereby any surface is treated as Lambertian emitter using values from the map.

II.8.3 PATH TRACING

Additionally to a *random walk* a NEE on each path vertex can be performed. This method was first applied by Kajiya [1986] and called path tracing. Kajiya's main observation was, that it suffices to follow a single path recursively, instead of branching the path at any intersection point. He did not yet consider the random hits and combinations of random hits and NEE.

In this thesis I will use the term *Path Tracing* (PT) for the rendering method which combines uPT and the connections to the light source using MIS. This is a common choice. Today, many industrial renderers perform variations of this kind of PT as main simulation method.



To combine the two operations connect and random hit we apply MIS. Since the balance heuristic is a special case of the power heuristic ($\beta = 1$) the following transformations are only shown for the power heuristic. To compute the weight from Equation (II.24) we set $N_{\text{rl}} = 1$ and insert the path densities p_{rv} and p_{c}. We let N_{c} be variable, because it is possible to perform multiple NEEs at each vertex.

Balance heuristic Eq. (II.23) p. 23 and power heuristic Eq. (II.24) p. 23

p_{rv} defined in Eq. (II.77) p. 66
p_{c} defined in Eq. (II.80) p. 67

$$w_{\text{rv}} = \frac{p_{\text{rv}}^{\beta}}{p_{\text{rv}}^{\beta} + (N_{\text{c}} p_{\text{c}})^{\beta}} = \frac{1}{1 + \left(\frac{N_{\text{c}} p_{\text{c}}}{p_{\text{rv}}}\right)^{\beta}} = \frac{1}{1 + \left(\frac{N_{\text{c}} p(\boldsymbol{x}_\ell)}{p(\boldsymbol{x}_\ell | \boldsymbol{x}_{\ell-1})}\right)^{\beta}} \tag{II.86a}$$

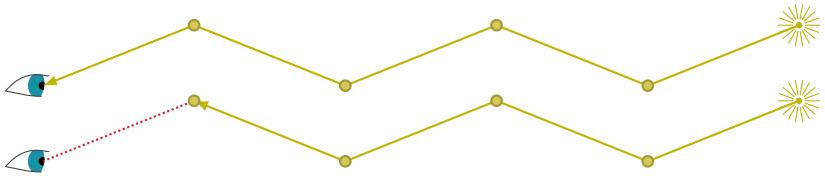
$$w_{\text{c}} = \frac{(N_{\text{c}} p_{\text{c}})^{\beta}}{p_{\text{rv}}^{\beta} + (N_{\text{c}} p_{\text{c}})^{\beta}} = \frac{1}{1 + \left(\frac{p_{\text{rv}}}{N_{\text{c}} p_{\text{c}}}\right)^{\beta}} = \frac{1}{1 + \left(\frac{p(\boldsymbol{x}_\ell | \boldsymbol{x}_{\ell-1})}{N_{\text{c}} p(\boldsymbol{x}_\ell)}\right)^{\beta}} \tag{II.86b}$$

The transformation into the double fraction is necessary in practice to increase the numerical robustness. Otherwise the sum of probabilities can

easily overflow the numeric range. In floating point this results in a finite number divided by infinity which would cause both weights to become zero. In the form on the right, identical terms between the random walk and the connection towards the last vertex are canceled, leaving only two basic probability densities.

Figure II.23 shows that PT has no problems with small light sources in combination with diffuse surfaces. It is still as bad as uPT for caustics and SDS paths. It will find all paths of the type L(DD|(SD|DS)D)(·)*E which means that for infinitesimal and small light sources there must be a diffuse vertex on the next surface.

II.8.4 Light Tracing



Analogously to path tracing, the last vertex of the light path can be connected to the sensor. This was first done by Dutré et al. [1993]. Like for PT I will use the term *Light Tracing* (LT) to refer to the combination of uLT and the connection to the sensor. However, in most cases where the pinhole camera is used, this means that only the connection is established.

If both contributions are computed, the MIS weights are derived like for PT resulting in

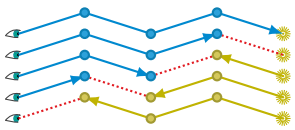
$$w_{\mathrm{rl}} = \frac{1}{1 + \left(\frac{p(\boldsymbol{x}_0)}{p(\boldsymbol{x}_0|\boldsymbol{x}_1)}\right)^{\beta}} \qquad w_{\mathrm{c}} = \frac{1}{1 + \left(\frac{p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{p(\boldsymbol{x}_0)}\right)^{\beta}}. \tag{II.87}$$

PT MIS weights Eq. (II.86) p. 73

The sample-able paths are L(·)*(DD|D(SD|DS))E which, opposed to PT, exchanges the variability of the light source with the variability of the camera model. As visible in Figure II.24, caustics and diffuse paths are captured well. On the downside, specular objects appear completely black.

II.8.5 Bidirectional Path Tracing

In the next step we can combine PT and LT. Moreover, it makes sense to allow connections at any point between two sub-paths. In *Bidirectional Path Tracing* (BPT) one light path is sampled for each view path. Then, every possible connection between the two paths is computed and tested for visibility. This algorithm was independently introduced by Lafortune and Willems [1993] and Veach and Guibas [1995a]. The differences between both are the heuristics to weight the numerous possibilities against each other. Veach investigated this weighting more deeply [Veach and Guibas 1995b] and found the afore introduced balance and power heuristic.



Balance heuristic Eq. (II.23) p. 23 and power heuristic Eq. (II.24) p. 23

The weights for BPT are derived the same way as for PT or LT including the double fraction trick for numeric robustness. However, this time it is

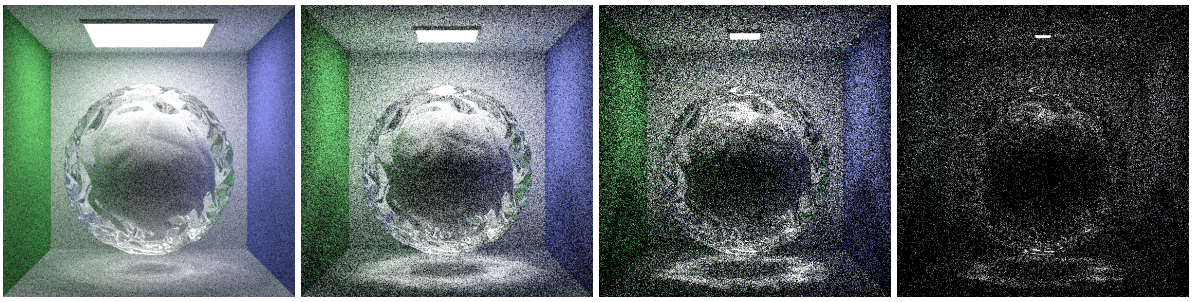


Figure II.22: Unidirectional Path Tracing (128spp). The variance increases the smaller the light source.

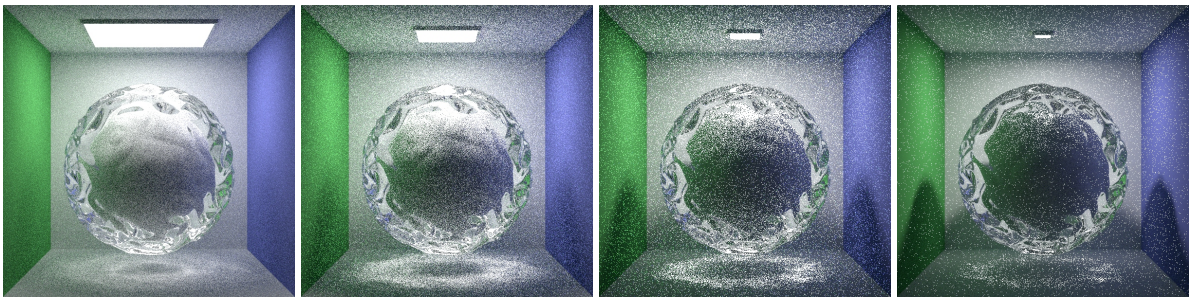


Figure II.23: Path Tracing (128spp). PT has the same problem with small light sources like uPT, although it is restricted to caustic and SDS paths.

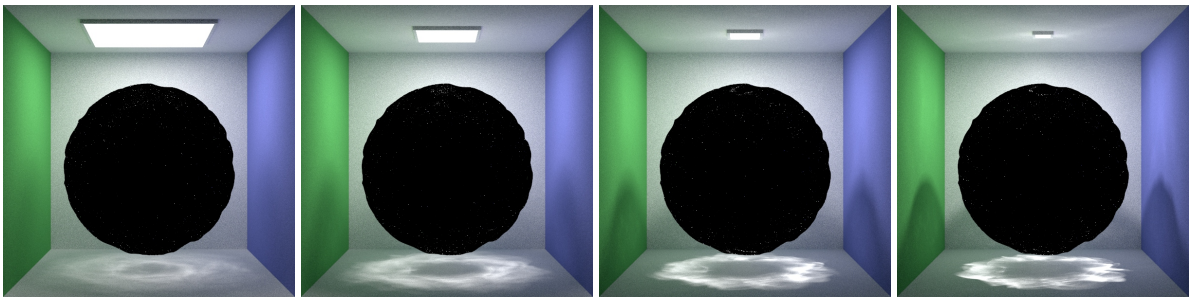


Figure II.24: Light Tracing (128spp). LT is able to handle all kinds of light sources well. Instead, it is not able to connect a smooth surface with a tiny camera.

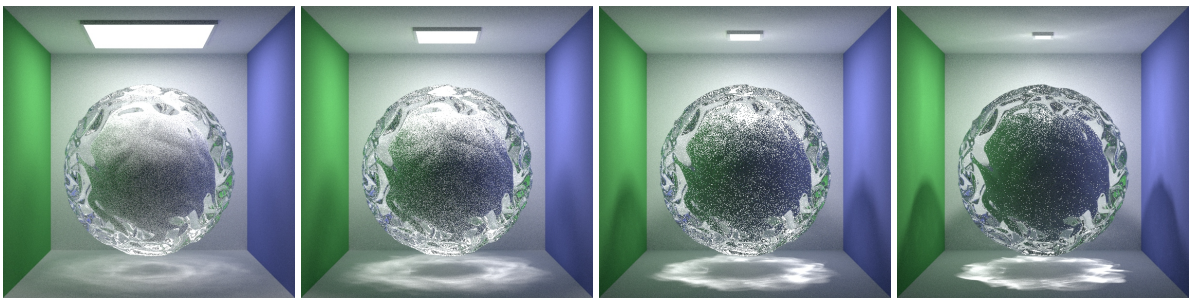


Figure II.25: Bidirectional Path Tracing (128spp). BPT combines the strengths of PT and LT. It is still not able to draw reflected and refracted caustics (SDS paths).

not possible to shorten as many terms as before. The weight

$$w_{\mathrm{c},i} = \left(\sum_{j=1}^{i} \left(N_j \prod_{k=j}^{i} \frac{p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})}{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})} \right)^{\beta} + 1 + \sum_{j=i+1}^{\ell} \left(N_j \prod_{k=i+1}^{j} \frac{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}{p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})} \right)^{\beta} \right)^{-1} \tag{II.88}$$

is computed recursively along the path in practice.

Each term $p(\boldsymbol{x}_k|\boldsymbol{x}_{k\pm1})/p(\boldsymbol{x}_k|\boldsymbol{x}_{k\mp1})$ moves the connection by one segment. It means we go one further random walk step on one path ($p(\boldsymbol{x}_k|\boldsymbol{x}_{k\pm1})$) and do one step less on the adjoint path ($1/p(\boldsymbol{x}_k|\boldsymbol{x}_{k\mp1})$). Beginning at the current sampler with $p_{\mathrm{rel},i} = 1$, we can compute the relative sampler probabilities recursively along both directions:

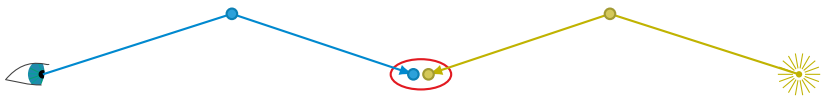
$$p_{\mathrm{rel},i\pm1} = p_{\mathrm{rel},i} \cdot \frac{p(\boldsymbol{x}_{i\pm1}|\boldsymbol{x}_i)}{p(\boldsymbol{x}_{i\pm1}|\boldsymbol{x}_{i\pm2})}.$$

The two sums in Equation (II.88) are sums over all possible $p_{\mathrm{rel},i}$ on the two sub paths. The right sum includes the random hit paths for $j = \ell$. Thereby, $p(\boldsymbol{x}_\ell|\boldsymbol{x}_{\ell+1})$ is identical to $p(\boldsymbol{x}_\ell)$ – the probability to choose the start vertex.

Simply said, the BPT is capable of rendering all paths with at least two adjacent diffuse vertices. Formally, this is a little more complex because orthographic lights/cameras and point sources/cameras can be treated well, if the next vertex is diffuse. This was also the case for the PT and LT whose paths can be both found by the BPT. Additionally, it finds paths with arbitrary end points L(·)*DD(·)*E, if there are two real diffuse vertices in between. BPT still fails for the SDS path, which only contains a single diffuse vertex, and the fully specular path. This can be observed in Figure II.25.

PT: L(DD|(SD|DS)D)(·)*E
LT: L(·)*(DD|D(SD|DS))E

II.8.6 PHOTON MAPPING



One operation we did not use so far is the *merge* event. The first algorithm using merges was the photon mapping by Jensen [1996]. The algorithm has two passes. First, the photons (light sub-paths) are traced and stored on diffuse surfaces. In a second pass, the view sub-paths are traced until they hit a diffuse surface at which the close by photons are collected. From the density of these photons the (ir)radiance can be computed which is called the *radiance estimate*.

Merges II.7.5 p. 68

There are alternatives in the definition of *close by* and in the used search data structures. Jensen used the *k-Nearest-Neighbor* (kNN) photons for the radiance estimate. The commonly used data structure to find the kNNs is the kd-tree from Bentley [1975]. To perform the *radiance estimate* one divides the summed contribution (flux times BSDF) by the area on which the k photons lie. If the used kernel is non-zero at is border, like the uniform one, it is correct to use the radius of $k+1$th neighbor to compute the area of a disc [García et al. 2012]. Otherwise one might use the radius of the kth neighbor, in which case its contribution becomes zero due to the kernel. Thus, it is always necessary to query the distance of $k + 1$ neighbors if a contribution over k elements should be averaged.

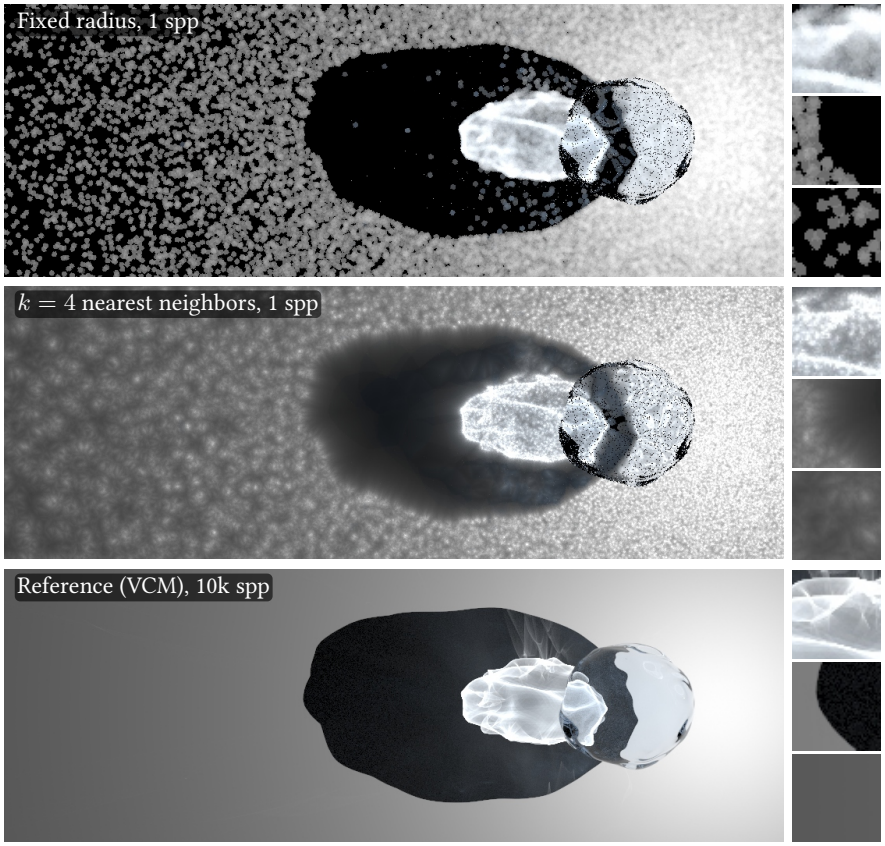


Figure II.26: Comparison of merge methods. In both cases a uniform kernel is used and the photon distribution is the same. The kNN estimates are less biased in caustics (top closeup) and have less variance in low density regions (bottom closeup). However, the stronger blur in dark regions may also increases the bias (middle closeup).

Much more work was done to reduce the bias of density estimates. For more details on general density estimates the book of Silverman [1986] or the survey of Sheather [2004] are good starting points. In the application of photon mapping several bias compensation techniques were proposed [Schregle 2003] and analyzed [Hernandez et al. 2014]. An overview of further merge kernels can be found in the photon mapping survey of Kang et al. [2016].

Other methods set the search region to a fixed size beforehand and estimate the density by summing all photons within the search region. Opposed to the kNN approach this increases the variance in low density regions and the bias in high density regions. In Figure II.26 this is demonstrated. However, as mentioned in Section II.7.5, it is important to know the merge area to compute MIS weights. This is not possible with the kNN approach without a previous search at each path vertex which is considered more expensive. Further, one often takes a merge radius which is smaller than the pixel's footprint such that the bias becomes unnoticeable.

Controlling the merge radius beforehand also allowed to introduce a consistent estimator. Hachisuka et al. [2008] maintained a statistic at the gathering point and reduced the merge radius such that the expected number of photons stays constant over the iterations. Later, he and Jensen [2009]

found that it is also valid to maintain a statistic over all paths in a pixel without fixing their gathering location. That means that instead of counting the true photons within a merge region it is possible to make queries at different locations with the average statistics of the history. Knaus and Zwicker [2011] showed that it is even possible to shrink the radius globally without any statistic. They used the series

$$r_{i+1}^2 = r_i^2 \cdot \frac{i+\alpha}{i+1} \tag{II.89}$$

with $\alpha \in (0,1)$ which leads to a consistent estimator as well. Thereby, using small α values leads to a faster shrinking and therefore more variance while larger α keep more bias. Setting $\alpha = 2/3$ yields the optimal asymptotic behavior for photon mapping, namely $\mathcal{O}(N^{-2/3})$ for convergence of the variance [Kaplanyan and Dachsbacher 2013a]. Kaplanyan and Dachsbacher also showed that the simpler series

Monte Carlo convergence $\mathcal{O}(N^{-1})$ Eq. (II.20) p. 22

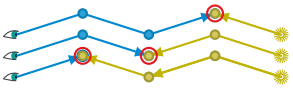
$$r_i = r_0 \cdot i^{\frac{\alpha-1}{2}} \tag{II.90}$$

has the same asymptotic behavior like equation (II.89).

Yet another approach to remove the bias from radiance estimates is the *Unbiased Photon Gathering* [Qin et al. 2015]. Qin et al. used another stochastic process to determine the unbiased contribution of a found neighbor. This technique is able to produce sharp caustics and shadow boundaries but involves a high non-constant cost per merge which makes it less applicable in GPU implementations.

II.8.7 Bidirectional Photon Mapping

In the original algorithm by Jensen [1996] there was no support to distinguish glossy surfaces. The tracing on both sub-paths proceeded on specular surfaces and stopped on Lambertian diffuse surfaces. Instead of a hard stopping criterion it is possible to use MIS weighting as in the previous methods. The resulting method is called *Bidirectional Photon Mapping* (BPM) and was introduced by Vorba [2011].



To derive the MIS weight we insert the path probability for merges p_m into the power heuristic. Thereby, it is necessary to use a fixed merge region as mentioned before. Only under this constant area assumption and with a uniform kernel, a simple MIS weight can be computed. We finally obtain

p_m Eq. (II.83) p. 68
Power heuristic Eq. (II.24) p. 23

$$w_{\mathrm{m},i} = \left(\sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \frac{p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)} \right)^\beta + 1 + \sum_{j=i+1}^{\ell-1} \left(\prod_{k=i+1}^{j} \frac{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}{p(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k)} \right)^\beta \right)^{-1} \tag{II.91}$$

which is very similar to the connection weight $w_{\mathrm{c},i}$ with two major differences. First, the compared probabilities between successive events are the two directions of the same segment opposed to adjacent segments as in BPT. Second, the number of possible events is smaller because there are no merges at the end points.

$w_{\mathrm{c},i}$ Eq. (II.88) p. 76

A merge is the only operation which is able to find the difficult SDS paths. Therefore, BPM is able to find paths of the form L(·)*D(·)*E. This is mightier than the previously introduced BPT, which required two successive diffuse

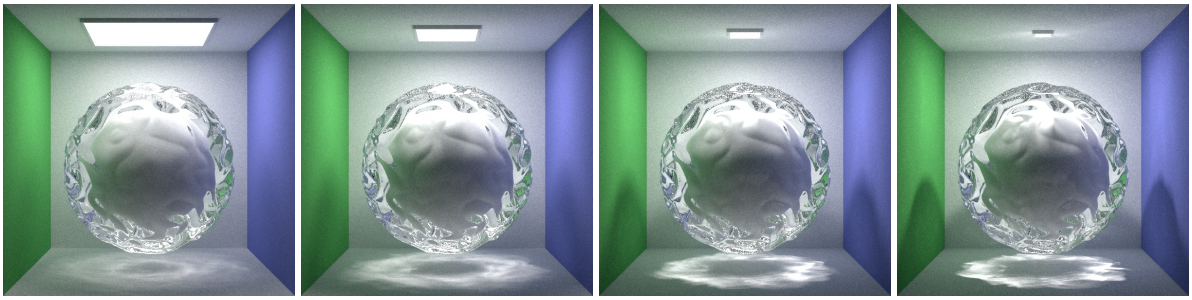


Figure II.27: Bidirectional Photon Mapping (128spp). Compared to methods in Figures II.22 to II.25, BPM is the first method which gets the SDS paths for smaller light sources with low noise.

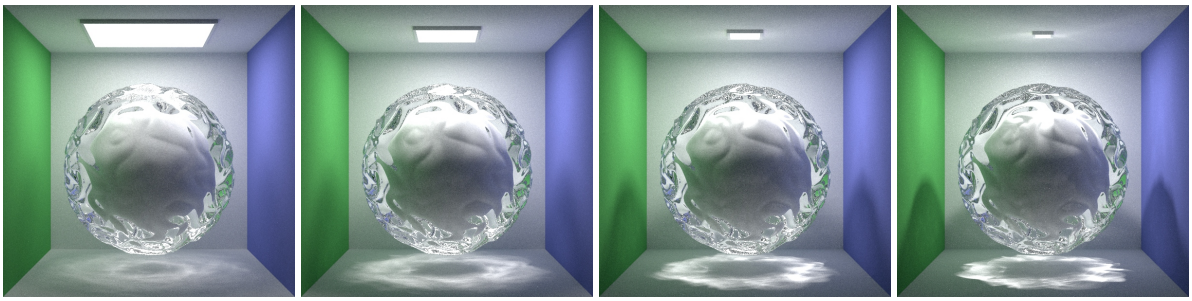
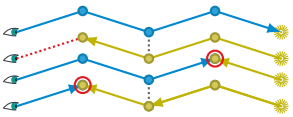


Figure II.28: Vertex Connection and Merging (128spp). Similar to BPM but with less noise and bias where the BPT part of the algorithm is successful.

vertices (L(·)*DD(·)*E). However, without the random hit operation BPM is not able to find the specular paths with a finite light source LDDS*E which was possible with uPT, PT and BPT.

In Figure II.27 we can see that BPM successfully renders the SDS paths. However, a direct comparison to BPT (Figure II.25) shows that BPM has more variance on the walls, where the NEE is the lower variance sampler.

II.8.8 Vertex Connection and Merging



Random hits II.7.3 p. 65
Connections II.7.4 p. 67
Merges II.7.5 p. 68

The last important step is to combine all the three sampling methods random hit, connection and merge into one method – the *Vertex Connection and Merging* (VCM). The difficulty in the combination is that connections have one less random event than merges. Simply adding the additional segment probability $p(\boldsymbol{x}_{k'}|\boldsymbol{x}_{k'+1})$ does not describe the difference between a merge at $\boldsymbol{x}_k/\boldsymbol{x}_{k'}$ and the connection between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$. Since the segment probabilities have the unit m^{-2}, scaling the scene would change the weight while the real sampler probabilities should not change.

Vertex k is the last of the view sub-path, k' the last of the light sub-path. For details see Eq. (II.83) p. 68

The solution is to include the merge area into the path probability and was published simultaneously by Georgiev et al. [2012] and by Hachisuka et al. [2012]. The point is that the larger the merge area the larger the probability to find the vertex $\boldsymbol{x}_{k'}$. The acceptance probability

$$p_{\text{acc}} = \pi r^2 p(\boldsymbol{x}_{k'}|\boldsymbol{x}_{k'+1}) \tag{II.92}$$

combines the per area density from the segment with the search region of the merge. That means p_{acc} is the expected number of photons which reach the merge region after the scattering at $\boldsymbol{x}_{k'+1}$. The necessary assumptions

are that r is known at each vertex of the path and that the uniform kernel is used. Note that r must not necessarily be the same at different vertices.

The same solution is obtained if we consider the power heuristic in its estimator form together with the throughput t_m. By shortening out the path measurement contribution function we obtain the merge path probability (Equation (II.83)) including the merge area, i.e. in its connection compatible form.

Power heuristic estimator form Eq. (II.84) p. 69
Merge estimator throughput t_m Eq. (II.81) p. 68

In the two VCM publications [Georgiev et al. 2012; Hachisuka et al. 2012] the topic of non-uniform kernels was treated differently. Georgiev et al. focused on p_acc as the random probability to find the merge. With that perspective, the kernel K becomes irrelevant since it does not change the probability for a merge. Hachisuka et al. kept K as long as possible and called it the probability density of a perturbation around $\boldsymbol{x}_i$. However, similar to the following discussion, they were forced to assume a constant kernel $1/\pi r^2$ to complete the proof $f/p \Rightarrow t_\mathrm{m}$.

Applying the estimator perspective $p_\mathrm{m} = f/t_\mathrm{m}$ we can better assess how a non-uniform merge kernel could be integrated. Referring back to the first line of Equation (II.83) we know

MIS weights with estimators Sec. II.7.6 p. 69

$$p_\mathrm{m} \propto \frac{1}{K(\boldsymbol{x}_k, \boldsymbol{x}_{k'})}.$$

This means smaller kernel values lead to larger p_m which in turn leads to a higher MIS weight. If the path throughput is scaled by a small value, then its absolute variance is small too. Thus samples at the boundary of the merge region would be preferred over close ones which increases the bias. This happens because MIS only optimizes for variance and ignores bias completely.

The bigger problem is that, for each other vertex, we must also compute a hypothetical merge path probability. This depends on the choice of $\boldsymbol{x}_{k'}$ which was called the perturbation by Hachisuka et al. So, we could randomly sample $\boldsymbol{x}_{k'}$ which would lead to noise in the MIS weight itself. Instead, we can approximate the result of the choice by computing the expected value with a uniform PDF over the merge region $\mathcal{A}$ around $\boldsymbol{x}_i$:

$$\begin{aligned}
\mathrm{E}[K] &\approx \int_{\mathcal{A}(\boldsymbol{x}_i)} K(\boldsymbol{x}_i, \boldsymbol{a}) \cdot p_\mathrm{uniform}(\boldsymbol{a}) \,\mathrm{d}\boldsymbol{a} \\
&= \frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|} \int_{\mathcal{A}(\boldsymbol{x}_i)} K(\boldsymbol{x}_i, \boldsymbol{a}) \,\mathrm{d}\boldsymbol{a} \\
&= \frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|} = \frac{1}{\pi r^2}.
\end{aligned}$$

The integral vanishes in the third line if the kernel is normalized, which it should be anyway to be consistent [Hernandez et al. 2014].

In summary, it is a good choice to use the uniform kernel assumption in the computation of MIS weights even if other kernels are used. It is the best guess we can make for non-merge vertices of our current path and it avoids the preference of more biased samples.

The final MIS weight in VCM must compute the relative sampler probabilities of all other possibilities. A path of length ℓ has ℓ connections, $\ell - 2$

merges and the random hit sampler:

$$w_{\mathrm{c},i} = \left(1 + \underbrace{\sum_{j=0,j\neq i}^{\ell} \left(N_{\mathrm{c},j}\frac{p_{\mathrm{c},j}}{p_{\mathrm{c},i}}\right)^{\beta}}_{\text{Other connections}} + \underbrace{\sum_{j=1}^{\ell-1} \left(N_{\mathrm{m},j}\frac{p_{\mathrm{m},j}}{p_{\mathrm{c},i}}\right)^{\beta}}_{\text{Merges}}\right)^{-1} \tag{II.93}$$

p_{c} Eq. (II.80) p. 67
p_{m} Eq. (II.83) p. 68

$$w_{\mathrm{m},i} = \left(1 + \underbrace{\sum_{j=0}^{\ell} \left(N_{\mathrm{c},j}\frac{p_{\mathrm{c},j}}{p_{\mathrm{m},i}}\right)^{\beta}}_{\text{Connections}} + \underbrace{\sum_{j=1,j\neq i}^{\ell-1} \left(N_{\mathrm{m},j}\frac{p_{\mathrm{m},j}}{p_{\mathrm{c},i}}\right)^{\beta}}_{\text{Other merges}}\right)^{-1} \tag{II.94}$$

The fractions of probabilities can be computed relatively as before. Typically all N_{c} are 1 and all N_{m} equal the number of light paths. This reflects the reuse of photons over all merge paths. Trying to merge only a single photon is much less efficient than comparable connection samplers due to the additional random event on the path. This was also demonstrated in the VCM paper [Georgiev et al. 2012]. Photon mapping becomes efficient through the fact that we always search for photons from all N_{m} light paths.

VCM is capable of rendering all paths from BPT and BPM: L(·)*D(·)*E and LDDS*E. Also, it should have less variance than BPT or BPM alone. However, it happens in some situations that VCM has a higher variance than BPT due to a systematic failure in the MIS weight computation, which is the topic of Chapter IV. Indeed, setting N_{m} to the number of light paths is not correct, because the effective reuse of the photons is smaller than this factor.

II.8.9 Other Methods

Besides the methods introduced in this section there are completely different sampling and light transport algorithms. On possible option is to replace the random walk with a *Markov Chain Monte Carlo* (MCMC) sampler. MCMC methods conditionally exchange paths based on a target function (often contribution or visibility). They form unbiased samplers, too, and are able to importance sample the target function in the long run. However, they often get stuck in local optimal and produce structured noise due to correlated samples. For this reasons they are not used for production rendering yet.

Among the best MCMC methods are the *Manifold Exploration Metropolis Light Transport* (MEMLT) from Jakob and Marschner [2012] and the *(Improved) Half Vector Space Light Transport* [Hanika et al. 2015b; Kaplanyan et al. 2014]. Šik et al. [2016] combined *Primary Sample Space MLT* [Kelemen et al. 2002] with VCM and obtained a very robust solution. For further information on MCMC please refer to the survey of Šik and Křivánek [2018].

Another family of global illumination algorithms are the *Radiosity* methods [Cohen et al. 1988; Greenberg et al. 1986]. They divide the scene into patches of purely Lambertian surfaces and compute the equilibrium state of light transport by solving a system of equations. While being noise free, they have limited applications due to patch discretization and restrictions for the materials. Although, it is possible to support non-Lambertian materials [Immel et al. 1986] the frequency is limited and further discretization artifacts are introduced. Having too many patches or discretized directions

leads to infeasible sizes of the equation system. Today, the main application of Radiosity methods is in the real-time applications where different iterative approaches are applied [Keller 1997; Laine et al. 2007; Martin and Einarsson 2010].

Chapter III
Data Structures for Density Estimation

There are many common strategies to get the density from a point cloud. This topic was already introduced in Section II.8.6 and visually compared in Figure II.26. However, in two of my papers I needed to query the local particle density under slightly different design targets which are:

Photon Mapping II.8.6 p. 76

Smoothness The variance of the query should be as small as possible

Performance Both building and querying should have as small as possible overhead. As will be shown in this section, the common strategies may lead to impractical execution times.

In the paper [Jendersie and Grosch 2018]

An Improved Multiple Importance Sampling Heuristic for Density Estimates in Light Transport Simulation
Johannes Jendersie and Thorsten Grosch
In: Proc. of Eurographics Symposium on Rendering EI&I Track, pp. 65–72

I used a spatial hash map to query the density at each existing vertex of a light transport simulation. This density is then used to modify the MIS weights in photon mapping related techniques like BPM or VCM. More on this topic will follow in Section IV.

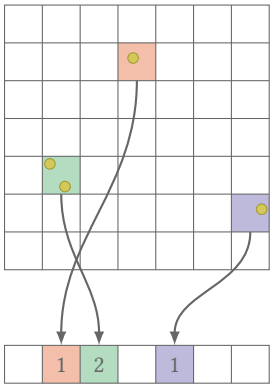
The hash map in this section has an improved robustness compared to that of the paper [Jendersie and Grosch 2018]. It properly handles hash collisions between cells and uses a more correct estimation of the density using the local area. Additionally, I introduce a robust interpolation scheme which is faster than the radial base function approach used in the paper.

In the paper [Jendersie 2019a]

Next Event Backtracking
Johannes Jendersie
In: arXiv:1909.00573

the requirements also included a less biased solution than is possible with the hash grid. I developed an octree-based variant which has less bias in high density regions and less noise in sparse regions. At the end of this section I compare the kNN query and the fixed radius query with the two data structures introduced here. Indeed, the octree qualitatively outperforms all others from the first iteration on, whereas the hash grid is superior to kNN after roughly 20 iterations.

III.1 A Hash Grid for Density Estimates



A hash grid is a uniform subdivision of the domain, where only cells containing data are stored. The sparsity is achieved by hashing the cell index as in usual hash maps. The advantages of a hash grid are its low memory footprint and its scalability to parallel hardware. I found that open addressing with quadratic probing worked exceptionally well in massive parallel implementations on the GPU. This is in accordance with the thesis of Alcantara [2011] who compared many different hash map implementations on the GPU.

To measure the particle density we can count particles per cell by storing a single integer. To measure the energy density we need to accumulate the path throughputs instead. In both cases the insertion algorithm is possible in atomic fashion. The following code snipped gives an overview of the atomic implementation.

```
type Entry
  u32[3] cell  # The position of the stored cell
  u32 count    # Particle counter or f32 to measure energy density
end

type DensityHashGrid
  Entry[] data    # Array with all stored cells (fixed size during
                  # runtime, set to a prime number in initialization)
  f32[3] cellSize # Spatial granularity of cells
end

func insert(DensityHashGrid grid, f32[3] pos)
  u32[3] cell = <u32>(floor(pos / grid.cellSize))
  # Simple hash similar to a linear congruential generator
  u32 hash = dot(cell, [0xb286aff7, 0x35e4a487, 0x75a9c18f])
  s = 0 # Use quadratic probing (begin with step 0)
  while true:
    idx = (hash + (s&1 ? s*s : -s*s) + len(grid.data)) % len(grid.data)
    expected = 0
    if atomic_cmp_swap(inout grid.data[idx].count, inout expected, ~0):
      # Cell had a 0-count before (unused), now it is marked with ~0
      # The marker locks the cell so we can initialize it properly.
      grid.data[idx].cell = cell
      grid.data[idx].count = 1  # release lock
    elif expected != ~0: # if not in allocation
      if grid.data[idx].cell == cell:
        atomic_add(inout grid.data[idx].count, 1)
        return
      end
      s += 1
    end # spin lock (do not change s and repeat loop)
  end
end
```

Collisions between cells are resolved by the quadratic probing. The necessary comparison is the only reason why each `Entry` must store the cell position. The main problem is that we want to allocate each cell exactly once. Thus we need to implement a spin-lock which lets all threads wait which try to insert into a location during its allocation.

Now, retrieving the counter for an arbitrary query point is a constant

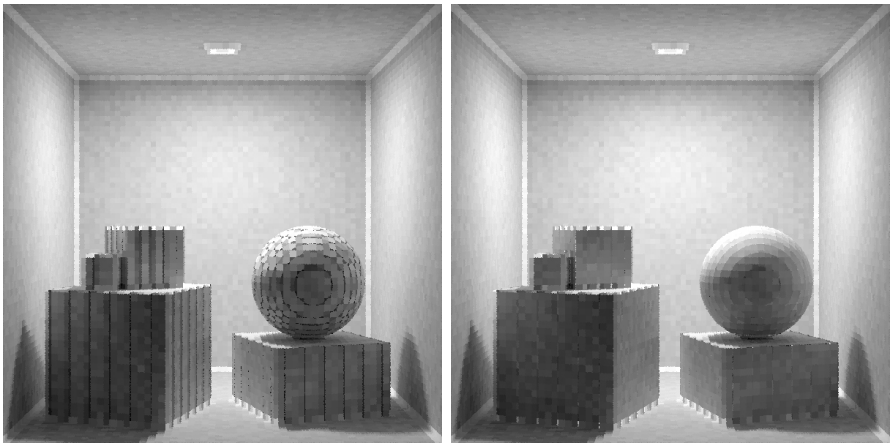


Figure III.1: Visualization of the counters in the density hash grid (left) and the density (right). Data is shown after tracing 5.76 million light paths (16 spp).

time operation on average[1]. A query must compute the same hash and follow the probing sequence until the cell itself or an empty cell is found. The result of such a query can be seen in Figure III.1 (left). Note that the result is a counter and not yet a density. Also, the grid architecture leads to aliasing patterns on the sphere and the boxes.

Both problems are resolved by dividing the counter by the intersection area between the scene and the respective grid cell. Unfortunately, it is difficult to obtain the correct area. Instead we can use a planar assumption and compute the intersection area between the tangential plane at the query point and the grid cell. The respective result is shown on the right of Figure III.1.

III.1.1 Intersection Area Plane / Box

It is possible to compute the intersection area between plane and box without determining the intersection itself. Although I am sure this formula is not new, I found no cite-able reference in the literature. The derivation here follows the Mathoverflow post https://math.stackexchange.com/a/885662/661978 from Achille Hui.

The basic idea is to compute the volume V inside the box and below the plane with respect to the plane offset $h = \langle \boldsymbol{n}, \boldsymbol{p} \rangle$. Then, the derivation of V yields the searched area. As primary conditions we have $\|\boldsymbol{n}\| = 1$ and that the box is axis aligned. Let

$$\Delta(h, \boldsymbol{x}) = \left\{ \boldsymbol{y} = (y_1, \ldots, y_d) \in \mathbb{R}^d : \langle \boldsymbol{n}, \boldsymbol{y} \rangle \leq h \wedge \forall k \in [1, d] : y_k \geq x_k \right\} \tag{III.1}$$

be the d-dimensional, axis aligned simplex, which starts at $\boldsymbol{x}$ as depicted in Figure III.2. The volume of this simplex can be computed from a product of its (axis aligned) edge lengths with

$$V(\Delta(h, \boldsymbol{x})) = \frac{\max(0, h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle)^d}{d! \prod_{k=1}^{d} n_k}. \tag{III.2}$$

$\boldsymbol{n} = (n_1, \ldots, n_d)$

[1] If the hash map is not too full.

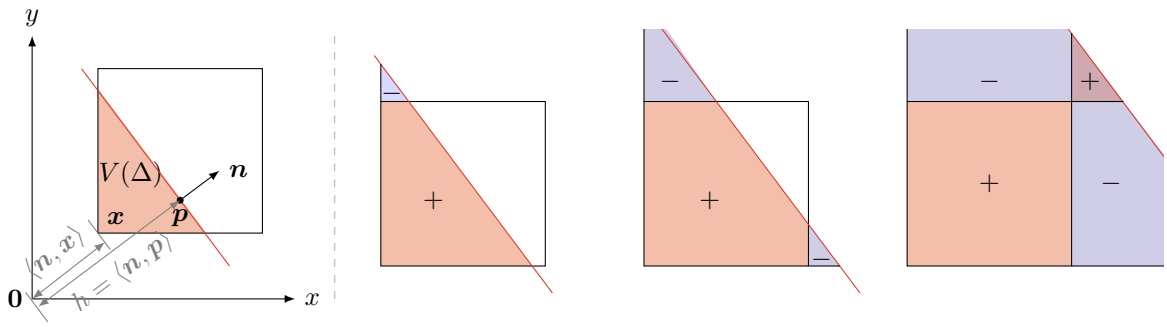
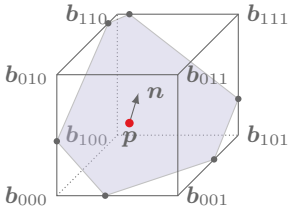


Figure III.2: Intersection area from simplex volume, shown for the 2D case where the simplex volume is the area of a triangle. Left: a single vertex of the box lies below the plane $(\boldsymbol{p}, \boldsymbol{n})$. Right: sequence of events when moving the plane along $\boldsymbol{n}$.

A derivation for the simplex volume can be found in Appendix A.3.2 p. 199. The clamping to zero is necessary if the respective simplex is completely on the upper side of the plane. In this case the intersection volume between simplex and half space must be zero.

Now, moving the plane along $\boldsymbol{n}$, it will pass all vertices of the box in a sorted order. This sequence is shown for 2D in Figure III.2, too. After passing the second vertex, the simplex from the first vertex will overestimate the volume. Here, the second simplex starting at the second vertex must be subtracted. The same applies to the third vertex. After the fourth vertex, the subtracted areas from the second and third vertex will overlap and the volume is underestimated. Thus, the volume of the fourth simplex must be added again. This alternating sign is described by the parity



$$\epsilon_i = \begin{cases} 1, & i \in \{000_2, 011_2, 101_2, 110_2\} \\ -1, & i \in \{100_2, 010_2, 001_2, 111_2\} \end{cases}$$

where the indices i are that of the eight box vertices in such an order that each bit toggles one of the dimensions. The same generalizes to arbitrary dimensions by counting the bits of the respective binary representations.

Hence, the volume of the box $\mathcal{B}$ with respect to the plane is

$$V(\mathcal{B}, h) = \frac{1}{d! \prod_{k=1}^{d} n_k} \sum_{i=0}^{2^d-1} \epsilon_i \max(0, h - \langle \boldsymbol{n}, \boldsymbol{b}_i \rangle)^d. \qquad \text{(III.3)} \quad \mathcal{B} = \{\boldsymbol{b}_0, \ldots, \boldsymbol{b}_7\}$$

Finally, the change of volume over h is dominated by the area of the infinitesimal slab which gives the area

$$\begin{aligned} A(\mathcal{B}, h) &= \frac{\partial V(\mathcal{B}, h)}{\partial h} \\ &= \frac{1}{(d-1)! \prod_{k=1}^{d} n_k} \sum_{i=0}^{2^d-1} \epsilon_i \max(0, h - \langle \boldsymbol{n}, \boldsymbol{b}_i \rangle)^{(d-1)}. \end{aligned} \qquad \text{(III.4)}$$

The final area for 3D is then obtained by applying the above equation for one, two and three dimensions. Each component of $\boldsymbol{n}$ which is zero means that the normal is perpendicular to the respective edge of the box. This would cause divisions by zero for the respective dimension. Instead we compute the projection of the remaining dimensions, which gives a simplex

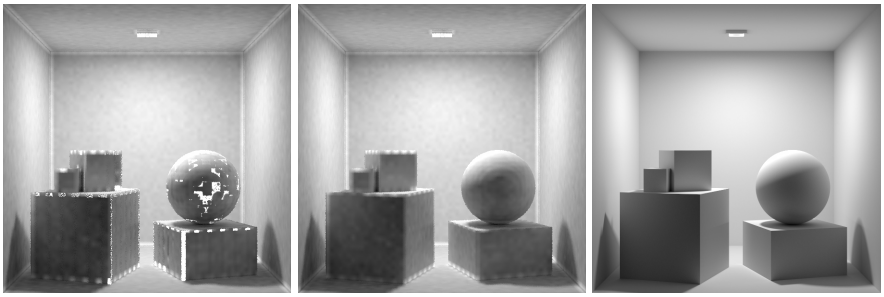


Figure III.3: Linearly interpolated data from Figure III.1. Left: Computing the density per cell before interpolation. Middle: Independent interpolation of count and area before the division. Right: Reference rendered with a modified LT

of a lower dimension. Then, the edge lengths of the reduced dimension must be multiplied with the area of the lower dimensional simplex. Let $\boldsymbol{s} = \boldsymbol{b}_{111} - \boldsymbol{b}_{000}$ be the size of the box:

$$A(\mathcal{B}, \boldsymbol{p}, \boldsymbol{n}) = \begin{cases} s_1 \cdot s_2, & n_1 = n_2 = 0 \\ s_1 \left| \frac{1}{n_2 n_3} \sum_{i=0}^{3} \epsilon_i \max\left(0, \langle \boldsymbol{n}, \boldsymbol{p} \rangle - \langle \boldsymbol{n}, \boldsymbol{b}_i \rangle\right) \right|, & n_1 = 0 \\ \left| \frac{1}{2 n_1 n_2 n_3} \sum_{i=0}^{7} \epsilon_i \max\left(0, \langle \boldsymbol{n}, \boldsymbol{p} \rangle - \langle \boldsymbol{n}, \boldsymbol{b}_i \rangle\right)^2 \right| \end{cases} \tag{III.5}$$

W.l.o.g. the sub-indices $\square_1$ and $\square_2$ in the first line and $\square_1$ in the second line are the dimensions for which the normal has a value of zero. For each of the two lines there are three possible combinations (xy, xz, yz and x, y, z) which are computed equivalently. The third line is the general case which applies to all non-axis-aligned normal vectors.

III.1.2 Hash Grid Interpolation

Although the area division fixed the problem of aliasing from the grid, we still have discretization discontinuities, as can be seen in Figure III.1 (right). The next logical step is to interpolate the values. This is simple with trilinear interpolation on the uniform grid. Instead of querying the counter from only one cell, we select the eight neighbors and compute the individual area for each of them.

Then, there are multiple possible ways to proceed. The most logical one is to divide each counter c_i of a cell i by its own area to get the density and to interpolate the eight density values. However, there will always be at least one cell which is not intersected by the tangential plane of the query point. This would lead to divisions by zero. So, all cells with an area of zero must be discarded and the weights must be renormalized after the interpolation. The interpolated density D is then

$$D = \frac{1}{\sum_i w_i \, H(A_i)} \sum_i w_i \, H(A_i) \frac{c_i}{A_i}. \tag{III.6}$$

Heaviside function

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Due to numerical problems in Equation (III.5), it still happens that invalid cells are included in the interpolation. The resulting artifacts are shown in Figure III.3 (left).

It would be possible to add some epsilon value to A_i before the division to avoid these artifacts. This would require a selection of an epsilon which is neither too small (artifacts) or too large (biased). A more robust strategy is to interpolate the counters and areas independently and to divide after summation:

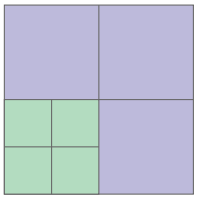
$$D = \frac{\sum_i w_i c_i}{\sum_i w_i A_i} \tag{III.7}$$

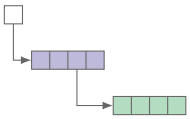
The result is shown in Figure III.3 in the middle. It is almost identical for most areas and does not contain the artifacts.

When comparing the results from the hash grid with a ground truth solution (Figure III.3 right) we see that the major remaining problem is topological bias around corners. Like for merges, the other types of bias may also happen. For example a caustic or shadow boundary would be blurred by the size of the grid cells. All kinds of bias can be reduced by decreasing the cell size. This, in turn, increases the noise level and memory consumption. A better solution would be to have an adaptive data structure which can be realized with a sparse octree.

Merge bias Sec. II.7.5

III.2 An Octree for Density Estimates





A hyper-octree subdivides the d-dimensional hypercube into 2^d equally sized child cells. If d is equal to two it is also called quadtree and if it equals three it is called octree. A sparse octree is an octree whose depth is adaptive to some scene-defined features. It cannot be stored implicitly like a perfectly balanced tree and thus must store its child pointers explicitly. A common strategy to keep the memory overhead for pointers at a minimum is to store the 2^d child nodes in consecutive blocks such that each node only needs a single child pointer.

To measure particle density we need an integer counter per cell. However, since we only need the counters on the finest level (in the leaves), it is possible to store the index of the first child instead of the counter in internal nodes. To encode this, we can store a negative number for the address and a positive one for a counter. Thus, the entire data structure can be represented in a single integer array.

Nevertheless, there are some fundamental problems to be discussed and solved. It is easiest to understand all the details by following a practical implementation. The first operation is the `insert` function which atomically increases the counter of a cell if it is not a pointer.

```
type DensityOctree
  i32[] data
  f32[3] sceneMin
  f32[3] sceneSize
  i32 splitThreshold # threshold to refine the octree
end

func insert(DensityOctree tree, f32[3] pos, f32[3] normal)
  offPos = pos - tree.sceneMin
  normPos = offPos / tree.sceneSize
  iPos = <i32>(normPos * (1 << 30))
  countOrChild = -1
  lvl = 1
  do:
    # Get the relative index of the child [0,7]
    gridPos = iPos >> (30 - lvl)
    idx = (gridPos[0]&1) + 2 * (gridPos[1]&1) + 4 * (gridPos[2]&1)
    idx -= countOrChild;  # 'Add' global offset (stored negative)
    countOrChild = increment_if_positive(tree, idx)
    countOrChild = split_node_cond(tree, idx, countOrChild, lvl,
                                   gridPos, offPos, normal)
    ++lvl
  while countOrChild < 0
end

# Atomic add if positive, returns the new value
func increment_if_positive(DensityOctree tree, i32 idx) -> i32
  oldV = tree.data[idx]
  do:
    if oldV < 0: return oldV # Do nothing, oldV is a child pointer
    newV = oldV + 1
  while atomic_cmp_swap(tree.data[idx], inout oldV, newV)
  return newV
end
```

It is necessary to use a spin-lock in `increment_if_positive` due to the encoding of addresses in negative numbers. However, I did not encounter

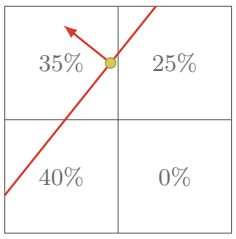
performance problems from this choice. Moreover, maintaining the two values in different arrays is slower due to higher cache miss rates. In one experiment the single array variant was around 5% faster.

The next open problem which needs to be solved is the splitting of cells. We want to split a cell if it lies within a high density region. This means we would like to have a constant number of particles per cell to balance variance and bias dependent on the local light situation. This again must be done in an atomic way.

If the counter is larger than some predefined threshold a split is necessary. To guarantee that only one thread will allocate a new node, we make the thread which reached the exact threshold responsible for the split. All others will have to spin-lock until the new node exists.

```
# Returns the child pointer or 0 to stop
func split_node_cond(DensityOctree tree, i32 idx, i32 count, i32 lvl,
                     i32[3] gridPos, f32[3] offPos, f32[3] normal) -> i32
  if lvl >= 30: return 0     # i32 grid resolution is limited.
  if count < 0: return count # Already a child.
  if count == tree.splitThreshold:
    if failed(newChildren = alloc_node(tree)):
      return 0               # Overflow
    init_children(newChildren, count, lvl, gridPos, offPos, normal)
    tree.data[idx] = -newChildren # Unlocks the other threads
  else
    # Lock until the responsible thread has initialized new nodes
    while 0 < (children = tree.data[idx]):
      if overflowed(tree): return 0 # Stop lock without a new child
    end
    return children
  end
end
```

Thereby the methods `alloc_node` and `overflowed` are trivial to implement by adding an atomic counter to the tree data structure. An allocation can be made by atomically increasing this counter and if the counter is larger or equal than `len(tree.data)` an overflow occurred.



Plane box intersection area
Sec. III.1.1 p. 85

A problem which needs more discussion is the initialization of new children. Since we do not have any information about the past insertions we cannot redistribute the information exactly. Instead we need to guess the distribution. The best approach I came up with is to use the current sample position and normal with the planarity assumption. With that plane we can compute the intersection area with all eight children and redistribute the count proportional to those areas. This, however, results in a very instable solution.

An obvious, alternative strategy is to distribute the count equally among all children. While it sounds intuitive, dividing the count by eight gives an underestimated result. The reason is that particles lie on a surface which intersects only parts of the eight cells. Therefore, distributing the amount equally in the volume is wrong. We get better results if a quarter count is assigned to each cell – according to the expectation that half of the cells will be empty.

In Figure III.4 all initialization strategies are visually compared. Note that independent of the initialization strategy, the bias from initialization will vanish over time. The solution on the very right is a hybrid between the two introduced strategies. At first, the count for a new cell is computed proportional to its intersection area. Then, it is clamped to the interval [old-

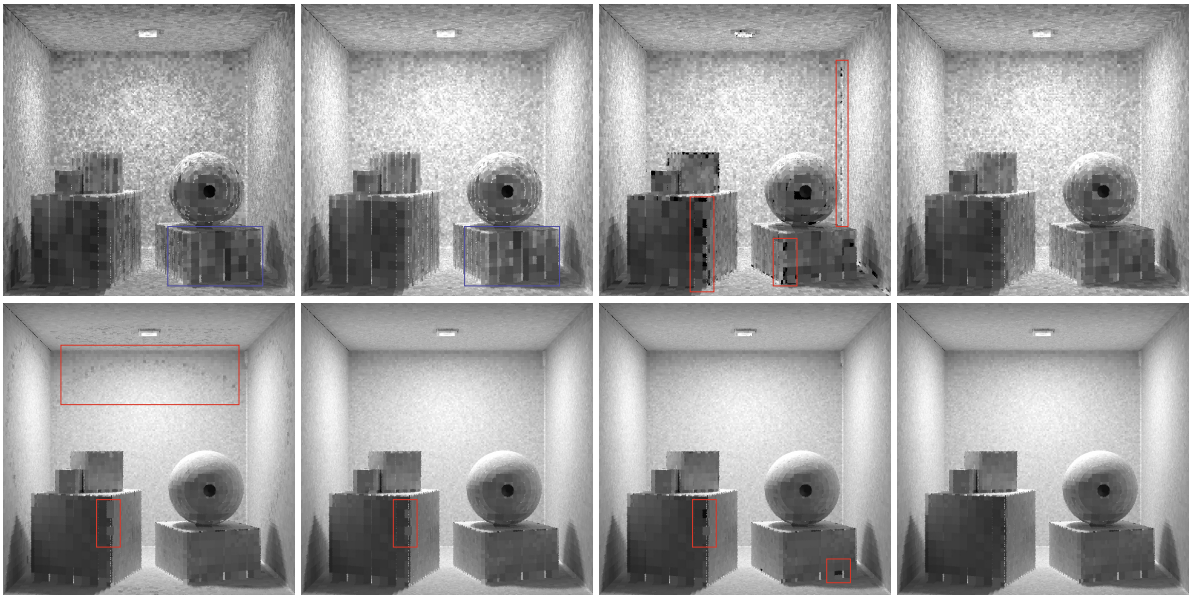


Figure III.4: Comparison of node initialization during splitting (Top 720k particles, bottom 11.5M particles). Fltr.: Uniform oldCount/8, uniform oldCount/4, area proportional and clamped area. The uniform variants show more stripe patterns (blue boxes). The pure area variant fails at corners, where the planarity assumption of the current sample does not hold. The clamped variant shows the least artifacts over any count of iterations.

Count / 8, oldCount - 1] to improve the robustness. For the application of clamping 'oldCount / 8' showed to be more correct than 'oldCount / 4' as in the uniform distribution variant. All following experiments and applications of the density octree will use the hybrid initialization strategy. The remaining artifacts in the images are due to the topological bias during the query. It can only be reduced by increasing the grid resolution (through decreasing the split threshold).

III.2.1 Interpolation in Sparse Octrees

Interpolation in sparse octrees is much more complicated than in uniform grids. The problem is that neighbors can lie on totally different levels. While octree interpolation occurs at some points in the literature, there is no satisfying solution.

A simple and promising method is the quadtree interpolation from the paper *Multiresolution Splatting for Indirect Illumination* [Nichols and Wyman 2009]. The idea is to upsample each individual level to the next higher resolution successively. After each upsampling step, all cells with a known value on that level are overwritten. I.e. the current grid is refined to twice the resolution and corrected with information from the existing nodes. However, an implementation of this approach showed severe ringing artifacts. The reason is that an interpolation on a high level can transport energy over a large distance, but local gradients on finer levels may go down to smaller values before the boundary to the interpolated region.

In realtime graphics this problem is circumvented by storing blocks with a higher resolution as leaf nodes – called bricks [P. H. Christensen and Batali 2004] – or by over-subdividing cells adjacent to query points to allow an easy interpolation [Crassin 2011, p. 43]. These methods lower the memory

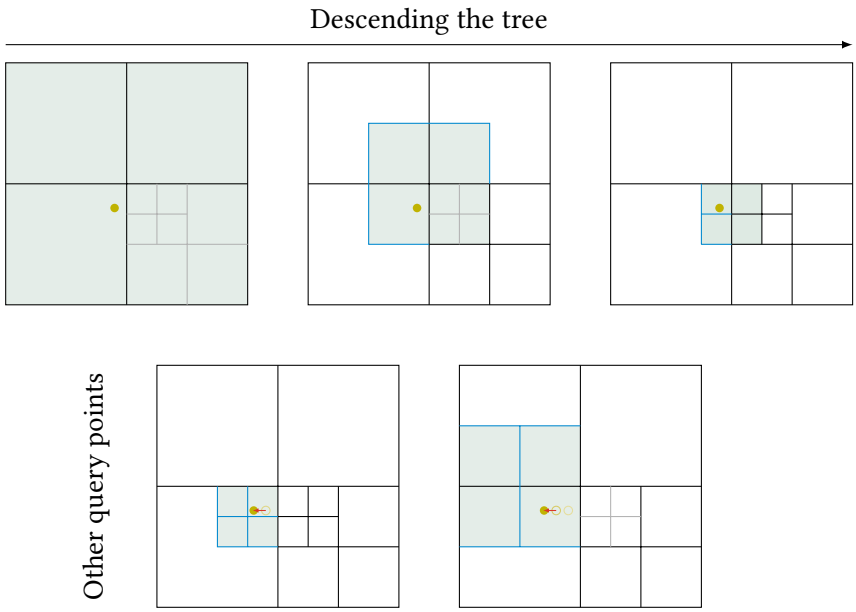


Figure III.5: Interpolation in an octree between real cells (black) and virtual ones (blue). For a given query point we descend the tree as long as any of the current cells has children. If the query point is moved it will eventually switch the level of interpolation (bottom right) which causes small discontinuities which are rarely visible.

efficiency and still have a problem if the resolution at the query point should be adaptive. They mainly focus on providing a high uniform resolution at surfaces while not storing information in empty volumes.

The method I ended up with is to track the eight cells for trilinear interpolation while descending the octree. The traversal proceeds as long as at least one of the eight cells has children. If we descend and a node has no children, its address and area are kept as is. It is then interpreted as a virtual child on the next level. Figure III.5 shows an example for different query points. Unfortunately, there are discontinuities when the level on which the interpolation takes place changes. However, these are barely noticeable in practice as is shown in Figure III.6.

At the end of the traversal, a trilinear interpolation of the eight cells can be performed. Note that the counter in virtual nodes was collected in a much larger region on some upper level. This must be correctly compensated. It is possible to compute the density, using the full sized node, first before the interpolation. Like in the hash grid this leads to severe problems with robustness.

Hash grid interpolation
Fig. III.3 p. 87

We can transfer the concept of interpolating counters and areas independently from the hash grid. This requires a normalization of the counter values in virtual cells, which is possible by dividing the area A_v of the virtual cell by the intersection area A_p of the original cell, leading to

$$D = \frac{\sum_i w_i c_i \frac{A_{v,i}+\varepsilon}{A_{p,i}+\varepsilon}}{\sum_i w_i A_{v,i}}. \tag{III.8}$$

The regularization with $\varepsilon = 10^{-2} A_{\text{avg}}$ is again necessary for numerical reasons where A_{avg} is the average area of the cell sides. Putting ε in both terms of the fraction makes sure that there is no error if $A_{p,i} = A_{v,i}$, so it

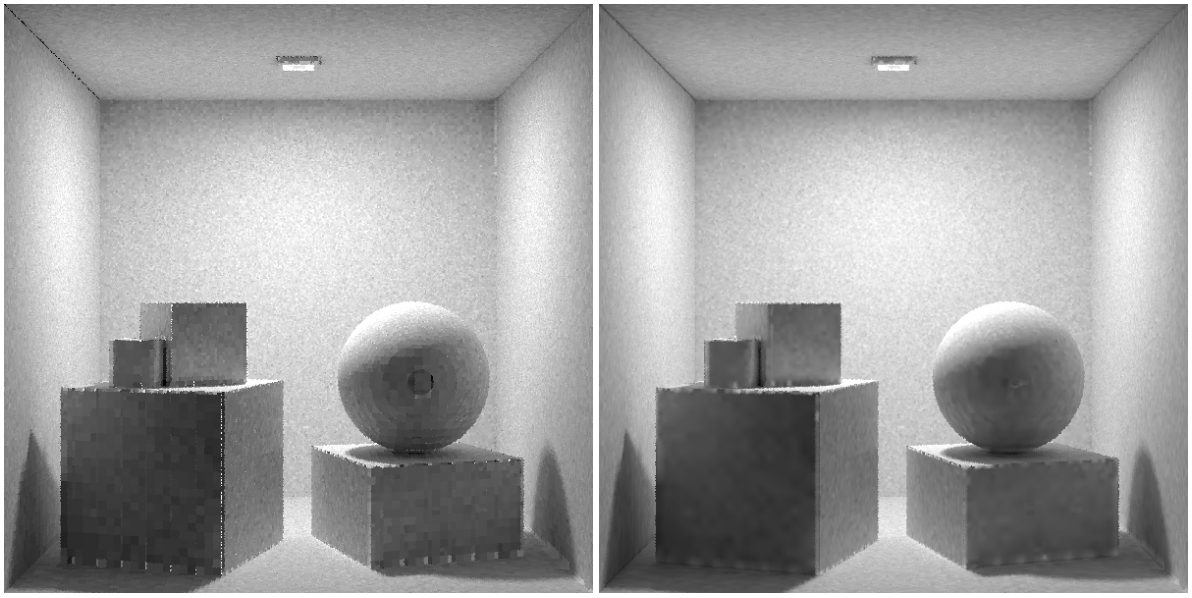


Figure III.6: Nearest sampling vs. linear interpolation with the proposed scheme (23M particles, split count 10 per iteration).

only affects virtual cells and does not need special casing for regular cells. Only if a cell is a virtual child $A_{p,i}$ and $A_{v,i}$ will be different.

The details in source code form can be found in Appendix A.4.1 p. 203.

III.2.2 Progressive Split Values

So far, all experiments were made with comparable high split values to emphasize the artifacts and maximize the smoothness. Also, it is not yet evaluated how the threshold should be set at all. To maintain a constant bias level, the threshold must be multiplied with the current iteration. Then, all iterations after the first one will only cause a few splits and all new data is used for variance reduction. On the other hand, it is possible to keep a constant threshold regardless of the iteration. In that case the variance level will stay roughly constant and each iteration will increase the depths of the tree. Any function between the constant and the linear increasing function will trade off the two errors in a progressive way.

The proposed implementation is robust enough to handle the inevitable memory overflow of the progressive strategy. Once there is no node left, the split threshold will be ignored and the integration will carry on in the existing cells. Thus, after a finite number of iterations the structure will remain constant and the variance in the cells will decrease afterwards.

In Figure III.7 several configurations are visualized. Choosing a large split threshold leads to smooth results starting with the first iteration (see first and third column). A small threshold leads to a lot more noise and has overestimation artifacts from the initialization bias (second column). Only with progressive refinement, which means we simply keep the threshold regardless of the number of distributed photons, bias reduces over time without having a bad start condition.

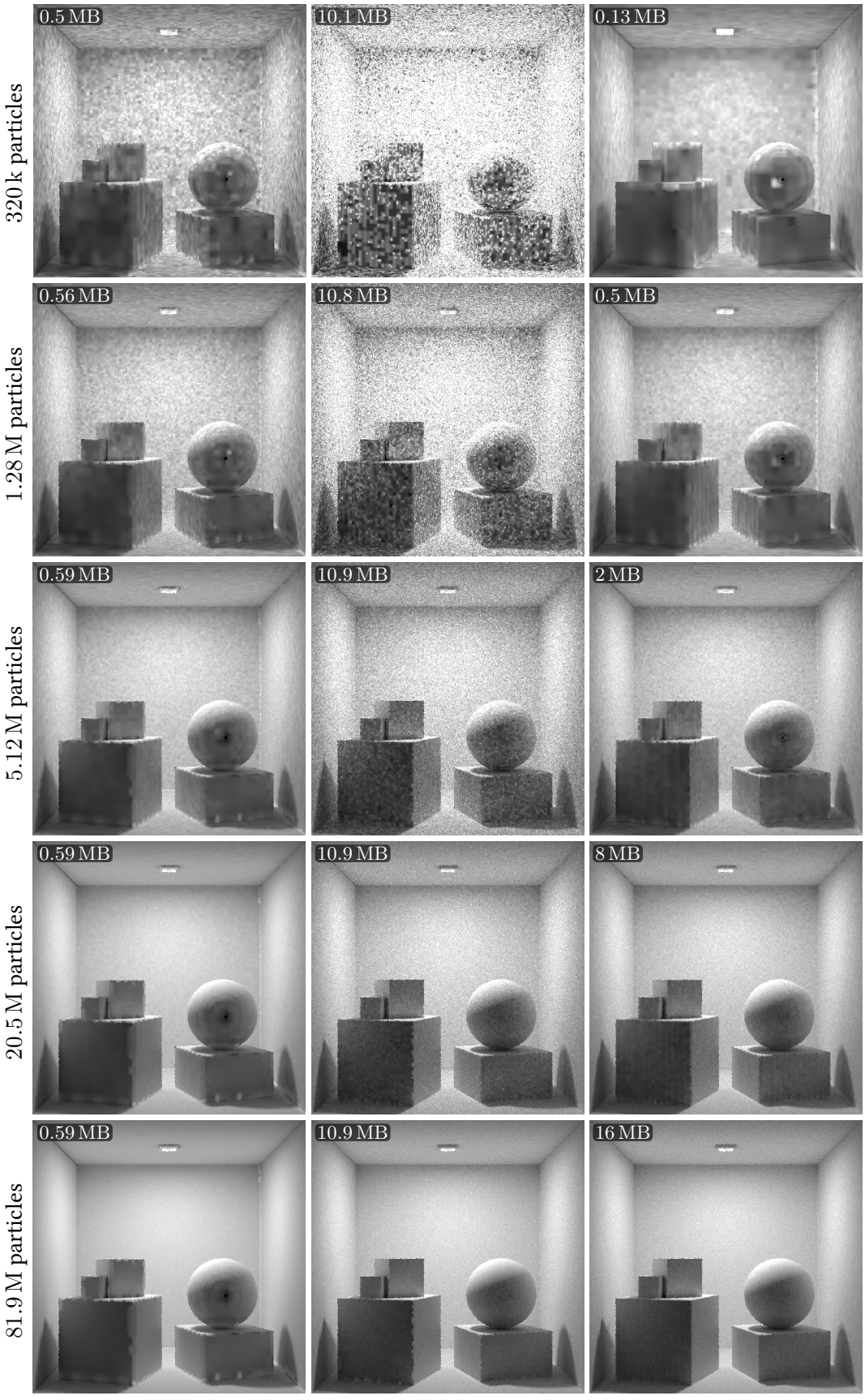


Figure III.7: Different threshold choices for node splitting. Memory is bounded to 16 MB. Left: threshold=16, growing per iteration. Middle: threshold=2, growing per iteration. Right: threshold=64, constant over iterations (progressive refinement, the memory boundary is hit at around 40 M particles after ≈128 iterations).

Table III.1: Query performance of different density estimates performed at each particle location.

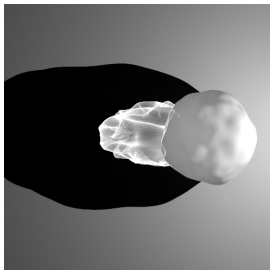
	kNN ($k = 5$)	Fixed Radius	Grid Nearest	Grid Interpolated	Octree Nearest	Octree Interpolated
Box	86.94 s	2058.99 s	4.46 s	23.29 s	6.94 s	47.51 s
Caustic	3237.98 s	92 075.64 s	1.40 s	5.52 s	1.55 s	11.57 s
Veach	75.23 s	50 463.78 s	4.30 s	21.89 s	9.44 s	99.25 s
Bath	72.06 s	144.68 s	4.04 s	22.34 s	6.70 s	48.93 s
Average	868.05 s	36 185.77 s	3.55 s	18.26 s	6.16 s	51.82 s

Table III.2: Build performance of different density estimate structures.

	kNN	Fixed Radius	Grid	Octree
Box	4.14 s	4.18 s	2.92 s	6.20 s
Caustic	0.75 s	1.41 s	0.60 s	1.11 s
Veach	5.06 s	3.86 s	2.58 s	9.39 s
Bath	3.45 s	3.85 s	2.59 s	6.55 s
Average	3.35 s	3.33 s	2.17 s	5.81 s

Box

Caustic



Veach

Bath

III.3 Comparison

In this section, four alternative approaches to query the density at each existing particle are compared. The first two are the kNN query and the fixed radius query which were introduced in the photon mapping section (II.8.6 p. 76). The other two are the hash grid and the octree which were introduced in this section.

All experiments are repeated for different scenes and the results are averaged. Both the hash grid and the octree are set up such that they use approximatively the same number of cells. For the octree I used a constant split threshold of 16 (progressive refinement) with 16 MB memory. The hash grid has a slightly larger memory footprint (18 MB). The small additional memory is required to avoid excessive collisions in a full map.

In the first experiment I measured performance by querying the density at each of the distributed particles (Table III.1). This is the scenario which will be the use case in the next section. In total, 10 M particles are distributed over 40 iterations. Additionally, the time to insert the data into the data structures is recorded (Table III.2)

The first thing to notice is that the fixed radius query requires huge amounts of time. The reason is that in high density regions many other particles lie within a search region. For each query all other particles are enumerated and, since many of the queries are performed in these same regions, the complexity of all queries together is quadratic in the density.

A similar artifact occurs in the kNN approach if there is a large empty region. Queries in low density regions often search large areas and therefore traverse many nodes of the tree.

Both newly designed data structures have a more stable performance. Without interpolation they outperform the other methods by far. However, enabling interpolation is expensive in both cases and the performance of

the octree comes close to that of a kNN search. The density hash grid is roughly 2–3× faster than the density octree.

III.3.1 ERROR RATES

Knowing the timings, it remains still open which method is the most efficient one. The efficiency is defined by the quotient of convergence rate and timings. Therefore, it is necessary to measure the error of the different methods. It is reasonable to use the visualization images and to compute an image error on them.

Most often the *Root Mean Squared Error* (RMSE)

$$\mathcal{E}_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - b_i)^2} \tag{III.9}$$

a_i, b_i Pixels of the images a and b

over all N pixels is used. However, I found that RMSE often performed poorly to compare Monte Carlo renderings. Since it takes absolute residuals $a_i - b_i$, errors in bright regions dominate the entire error.

An example: Assume a rendering where the error is a 1% underestimation in every pixel. In that image, 90% percent are dark regions with a value around 1. The remaining 10% are much brighter (caustic or directly lit) and have a value of 100. The error of that image would be

$$\mathcal{E}_{\text{RMSE}} = \sqrt{0.9 \cdot 0.01^2 + 0.1 \cdot 1^2} \approx 0.3164.$$

Now, we improve the algorithm and make the dark 90% completely free of any error. Then the measured error is

$$\mathcal{E}_{\text{RMSE}} = \sqrt{0.1 \cdot 1^2} \approx 0.3162.$$

So, by fixing 90% of the image we got an error improvement in the fourth significant decimal place.

Another example are fireflies. Those are even more extreme cases of the above example, where a single pixel causes the entire measured error. Both cases happened frequently in my experiments, so I moved on to the *Root Mean Squared Relative Error* (RMSRE)

$$\mathcal{E}_{\text{RMSRE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{(a_i - b_i)^2}{(a_i + b_i/2)^2}} \tag{III.10}$$

Def. RMSRE as a relative error measure

Dividing by the average of both pixel values is necessary to make the measurement symmetric. An asymmetric error, for example by dividing with image b only, would still cause problems for certain situations. Imagine a white pixel a_i on a black ground b_i which would cause an infinite error value. The other way around with a black pixel on white ground would only cause a small finite error. By taking the average, both dark and bright noise values are handled in the same way. This also reduces the dominance of fireflies considerably.

In the running example, we get more meaningful values this time. Before the hypothetic modification of error-free dark areas we have

$$\mathcal{E}_{\text{RMSRE}} = \sqrt{0.9 \cdot \left(\frac{0.01}{0.995}\right)^2 + 0.1 \cdot \left(\frac{1}{99.5}\right)^2} \approx 0.0101$$

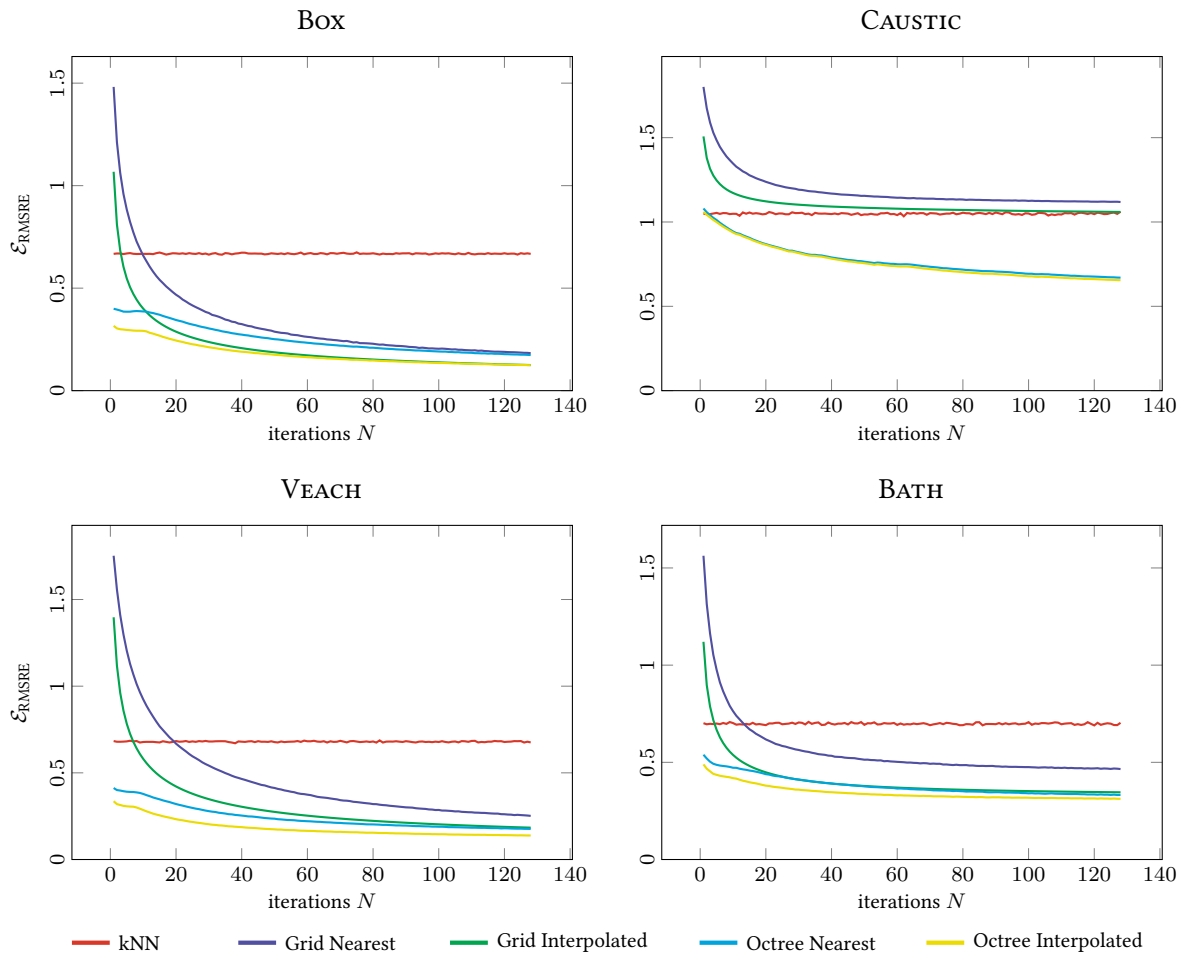


Figure III.8: Error plots for the different density query strategies. The reference images where created with the modified light tracer and are shown on page 95. The approaches using the new data structures remember all of the past and converge by themselves. The kNN based queries are equally good in each of the iterations since only photons of the current iteration are known.

and afterwards

$$\mathcal{E}_{\text{RMSRE}} = \sqrt{0.1 \cdot \frac{1}{99.5}^2} \approx 0.0032.$$

The result with $\mathcal{E}_{\text{RMSRE}}$ matches the observed improvements of light transport methods much better that of $\mathcal{E}_{\text{RMSE}}$. I have used the relative error in many experiments over the last years and always found it plausible. Indeed it behaves very similar to the more complex, perception-based *Structured Similarity* measure (SSIM) [Wang et al. 2004].

Going back to the actual topic we apply RMSRE on a series of density visualizations in the four test scenes. The plots of these series are given in Figure III.8. The fixed radius method is not shown, because an evaluation would have taken too much time. It definitely is the least efficient method. The settings are the same as for the performance experiments. In case of the kNN approach the density is estimated using the kernel

$$K(\boldsymbol{x}_k, \boldsymbol{x}'_k) = \frac{3}{\pi r^2}\left[1 - \left(\frac{\|\boldsymbol{x}_k - \boldsymbol{x}'_k\|}{r}\right)^2\right]^2 \qquad \text{(III.11)}$$

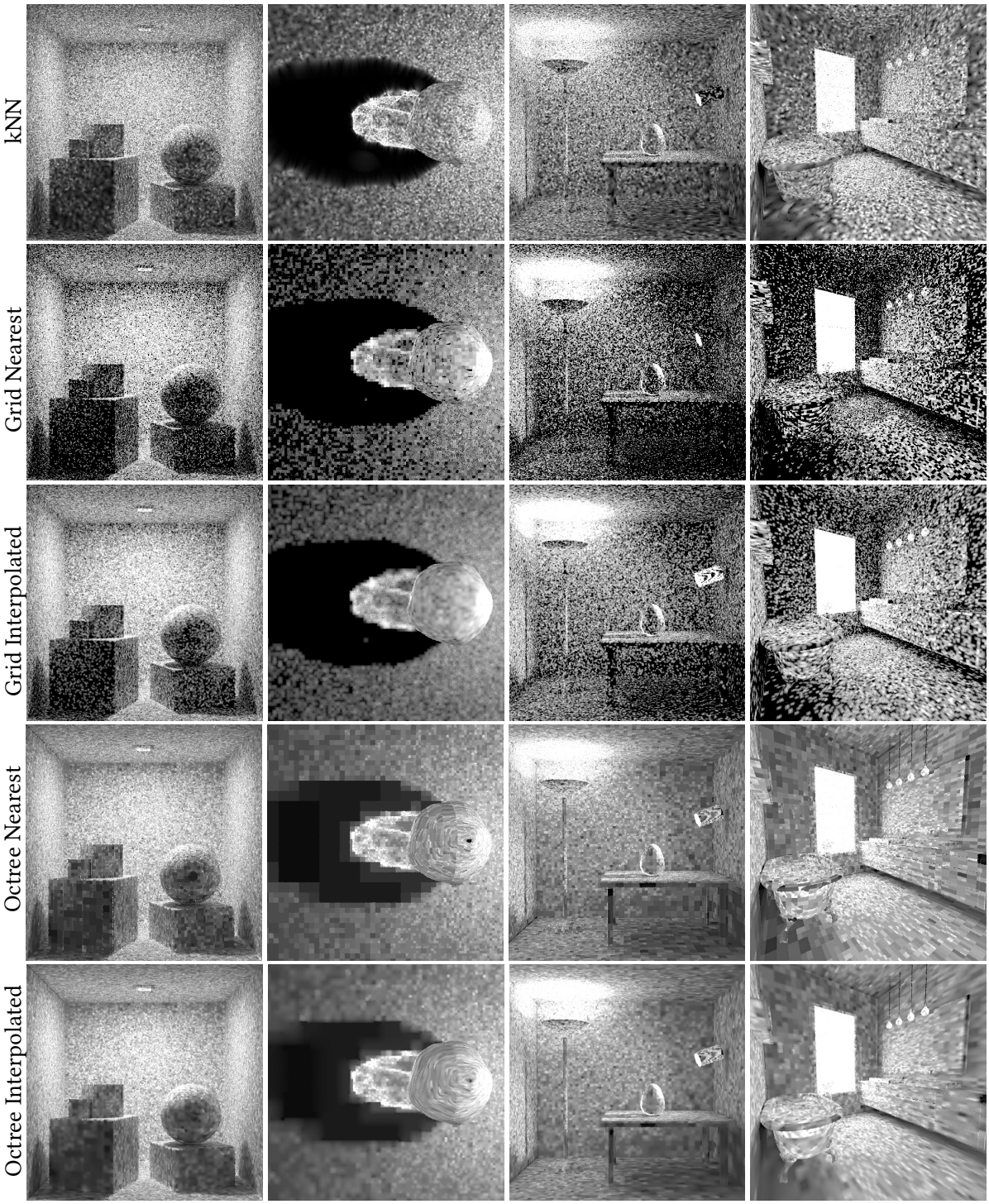


Figure III.9: Example query images for the first iteration.

which Schregle [2003] found to produce small bias results.

The plots show that the interpolated variants clearly have a smaller error compared to nearest sampling in the same data structure. The lines of the kNN approach are constant because, other than the dedicated structures, only the information of the current iteration is available. Keeping all particles from all iterations would simply exhaust the memory. The kNN density estimates are superior to the hash grid for the first few iterations only, except in the caustic scenario where the bias in the grid stays too large. In all cases the octree outperforms both other methods.

In the example images of Figure III.9 we can also see what kind of error is produced. In all methods the largest problem over time is the topological bias. The most severe cases are the cylindrical light source in the Veach scene and the tub in the bath. In both cases density bleeds through a thin structure due to insufficient grid subdivision (grid/tree) or because the closest neighbors are on opposite sides (kNN). At least in the first iteration, the noise in the hash grid is higher than in each other method which explains its high starting error. The kNN method balances the artifacts well, but is also more noisy than the octree.

From all used approaches only the octree can overcome the topological bias over time, if used in progressive configuration. To improve the octree's quality from the first iteration on, there is another solution candidate: Initializing the subdivision based on a frequency analysis of the local geometry in each cell. That means we can subdivide the octree in regions of high normal deviation before inserting any particle. However, any experiment in that direction failed to improve the robustness reliably.

Chapter IV
Improved MIS Heuristics Respecting Path-Reuse

This chapter summarizes the contributions made in my two publications [Jendersie 2019b; Jendersie and Grosch 2018] regarding the MIS failure in photon mapping methods:

An Improved Multiple Importance Sampling Heuristic for Density Estimates in Light Transport Simulation
Johannes Jendersie and Thorsten Grosch
In: Proc. of Eurographics Symposium on Rendering EI&I Track, pp. 65–72

Variance Reduction via Footprint Estimation in the Presence of Path Reuse
Johannes Jendersie
In: Ray Tracing Gems 1st ed., Edited by Eric Haines and Tomas Akenine-Möller, pp. 557–569

IV.1 The Problem with Path Reuse

As described in Section II.8.8, VCM combines the sampling methods in BPT and photon merges. Since none of the BPT samplers is discarded this should always keep or reduce the variance. There should always be a reduction of the variance if both merges and connections are able to sample a certain light effect. Nevertheless, did the first scene I rendered with VCM have an unexpected high variance.

VCM II.8.8 p. 79

Figure IV.1 demonstrates this case in the Veach-Bidir scene. The paths which cause the noise in this case are purely diffuse and should not lead to such problems. The cause is an overly large MIS-weight for the merges close to the light source.

The derivation of the weight was made for a single merge path [Georgiev et al. 2012; Hachisuka et al. 2012], for which it works as intended. Then, photons are reused over all paths and the total number of photons N_Φ is inserted into the power heuristic. However, *this is wrong, because only the light-sub paths are reused and not the entire sampler.* This is a fundamental problem with the power and balance heuristic. Since each path density is a product of several terms, it cannot be distinguished if only a part of this sampler is repeated.

Power Heuristic Eq. (II.24) p. 23

The arising questions is: how much is the variance reduced, if we reuse

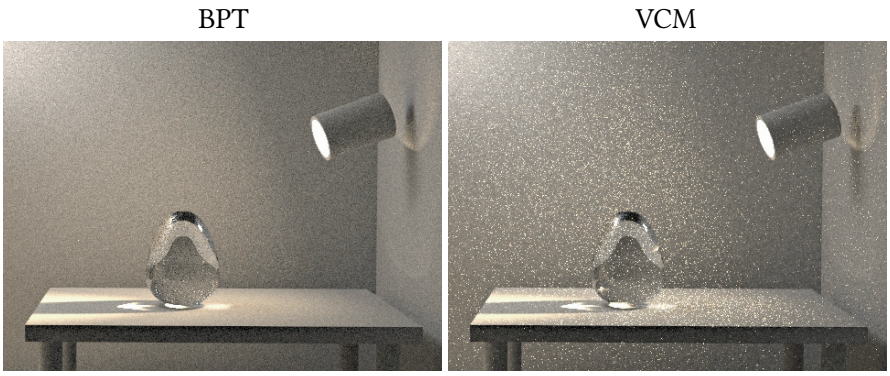
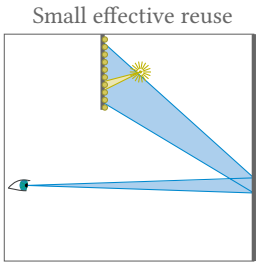
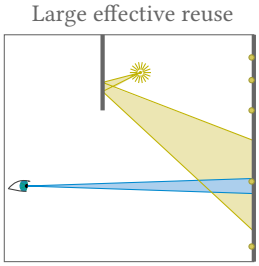


Figure IV.1: Failure case for VCM. Although it only adds samplers to the BPT method, it shows a higher variance. Both images have 50 spp.

one sub-path with a count of N_Φ? As a result we search a new effective reuse factor $N_R(\mathcal{P})$, which describes the gain more correctly. Note that N_R depends on the path $\mathcal{P}$ and is different for distinct paths. About the true reuse factor N_R we know a few special cases:

- $N_R \leq N_\Phi$: Repeating a part cannot be more effective than reusing the entire sampler.
- $N_R \geq 1$: Repeating a part of the sampler cannot be worse than the single sample value.
- $N_R = 1$ if the light sub-path causes no variance at all (e.g. deterministic reflected laser beam)
- $N_R = N_\Phi$ if the view sub-path causes no variance (does not happen in practice)





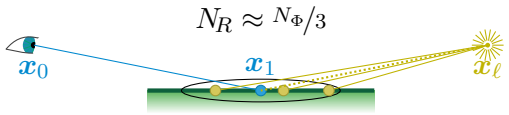
To compute the factor N_R, I took two different perspectives. The first was to count the number of correlated photons [Jendersie and Grosch 2018]. If the probability to observe another photon with the same path inside the current merge radius is high, the variance of light sub-paths is small. Or with other words: all correlated photons do not add additional information and as such do not reduce the variance. This will be explained in more detail in the next section (Sec. IV.2).

The second approach is based on direct observations of the variance [Jendersie 2019b], although I only came up with a weak heuristic to measure the variance of the two sub-paths. This second approach satisfies the above constraints without modifications. Further, it is faster to compute and needs less memory than my first proposed solution.

IV.2 Path Reuse Variance as a Consequence of Correlation

A merge assumes that two different sub-path end points are the same. Besides the bias which was discussed earlier, this also bends paths together such that they become identical. Consider the example of direct lighting. All photons falling into the merge radius will end up in the same path $\mathcal{P} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_\ell\}$. Therefore, repeating the photon sampler may generate correlated paths.

Merge bias II.7.5 p. 68



With this perspective the question is: How many paths will be identical after the merge? Let $N_{\equiv}(\mathcal{P})$ denote the number of correlated photons for a given path, then $N_R(\mathcal{P}) = N_\Phi / N_{\equiv}(\mathcal{P})$ tells us how many different paths we find through the reuse. Note that neither $N_{\equiv}$ nor N_R are necessarily integral numbers.

More formally, we can express the variance of correlated merge contributions t_m as

Merge contribution t_m Eq. (II.81) p. 68

$\mathrm{V}[X+Y]$ Eq. (II.16a) p. 20

$$\begin{aligned}
\mathrm{V}\left[\frac{1}{N_\Phi}\sum_{i=1}^{N_\Phi} t_{\mathrm{m},i}\right] &= \frac{1}{N_\Phi^2}\left(\sum_{i=1}^{N_\Phi}\mathrm{V}[t_{\mathrm{m},i}] + \sum_{i=1}^{N_\Phi}\sum_{\substack{j=1\\ j\neq i}}^{N_\Phi}\mathrm{Cov}[t_{\mathrm{m},i}, t_{\mathrm{m},j}]\right) \\
&= \frac{1}{N_\Phi^2}\left(N_\Phi \mathrm{V}[t_\mathrm{m}] + N_\Phi(N_\Phi - 1)\mathrm{E}[\mathrm{Cov}[t_{\mathrm{m},i}, t_{\mathrm{m},j}]]\right) \\
&= \frac{1}{N_\Phi}\mathrm{V}[t_\mathrm{m}] + \frac{N_\Phi - 1}{N_\Phi}\mathrm{E}[\mathrm{Cov}[t_{\mathrm{m},i}, t_{\mathrm{m},j}]]\,, \qquad \text{(IV.1)}
\end{aligned}$$

where $\mathrm{E}[\mathrm{Cov}[t_{\mathrm{m},i}, t_{\mathrm{m},j}]] = \overline{\mathrm{Cov}}_\mathrm{m}$ is the average covariance between samples. This means that, if all samples are fully correlated ($\mathrm{V}[t_\mathrm{m}] = \overline{\mathrm{Cov}}_\mathrm{m}$), the variance will remain constant regardless of the number of samples. In other words, additional correlated samples do not reduce the variance as effectively as uncorrelated ones. Let β be the ratio $\overline{\mathrm{Cov}}_\mathrm{m}/\mathrm{V}[t_\mathrm{m}]$, then

$$\text{(IV.1)} = \mathrm{V}[t_\mathrm{m}]\frac{1 + (N_\Phi - 1)\beta}{N_\Phi} \overset{\text{def.}}{=} \mathrm{V}[t_\mathrm{m}]\frac{N_{\equiv}}{N_\Phi} \overset{\text{def.}}{=} \frac{1}{N_R}\mathrm{V}[t_\mathrm{m}] \qquad \text{(IV.2)}$$

If we knew $\mathrm{V}[t_\mathrm{m}]$ and $\overline{\mathrm{Cov}}_\mathrm{m}$, we could compute β and $N_{\equiv} = 1 + (N_\Phi - 1)\beta$. However, it is not possible to obtain these values from a single given sample. Instead we are going to approximate $N_{\equiv}$ in the next step.

IV.2.1 VCM$^+$

Let $\overline{k}$ be the average number of photons found in the region of the current merge. This number is an upper bound for the number of correlated photons $N_{\equiv} \leq \overline{k}$ because there will never be more correlated samples than samples.

My first improved heuristic for the true reuse factor at vertex i

$$N_{R,i}^{+} = \frac{N_\Phi}{\max(1, \overline{k}_i)} \tag{IV.3}$$

makes direct use of this number [Jendersie and Grosch 2018]. The heuristic $N_\Phi / \overline{k}_i$ does not meet the enumerated conditions for N_R by itself. Especially, it can happen that $N_{R,i} > N_\Phi$ if the local density of photons is small ($\overline{k}_i < 1$). To fix this shortcoming, heuristic (IV.3) clamps the value of $\overline{k}_i$ to one. The VCM$^+$ heuristic already fixes the most severe failure cases of VCM, but is worse than the original for some other situations. This will be shown later in a joint evaluation section over all proposed solutions.

Evaluation Sec. IV.4 p. 119

IV.2.2 VCM*

It is possible to improve the previous heuristic with a simple normalization. We found experimentally that computing the harmonic mean of $1/\max(1, \overline{k}_j)$ over all possible merges j along the path gives a more robust solution. Through the normalization, the ratio between the summed merge and connection MIS-weights remains the same as in VCM. However, the weights within the merges are shifted toward the less correlated ones.

The final reuse factor for VCM* is

$$N_{R,i}^{*} = \frac{N_\Phi}{\max(1, \overline{k}_i)} \left(\frac{1}{\ell - 2} \sum_{j=1}^{\ell-1} \frac{1}{\max(1, \overline{k}_j)} \right)^{-1}. \tag{IV.4}$$

It is superior to the original VCM with full reuse count $N_R = N_\Phi$ for a high number of scenes and material configurations as will be shown in the evaluation section IV.4.

IV.2.3 Implementation and Costs

The two heuristics (IV.3) and (IV.4) make use of the average number of photons $\overline{k}_i$ in the merge region around $\boldsymbol{x}_i$. This is again an integrated number which is not available directly. In the original implementation [Jendersie and Grosch 2018], I used a density hash grid to integrate and query this value over the rendering process. The expected number of photons is

Density hash grid III.1 p. 84

$$\overline{k}_i = \pi r^2 D(\boldsymbol{x}_i) \tag{IV.5}$$

which is the merge area multiplied with the density D.

Density D from:
Hash grid Eq. (III.7) p. 88
Octree Eq. (III.8) p. 92

However, the original implementation had a much lower quality than the structures introduced in Section III.1 and III.2. It did not resolve hash collisions between different cells and did not use the intersection area to compute the density. Both together resulted in grid artifacts which were barely visible, but added to the error of the results. Visible errors mainly occurred through light-bleeding, if a grid cell spread over a thin wall.

Plane / Box intersection area III.1.1 p. 85

In this thesis both heuristics are re-evaluated using the much higher quality octree from Section III.2. Thus, the additional costs for VCM$^+$ and VCM* are the cost of maintaining and sampling the density octree. This adds around 16 MiB data and increases rendering time by roughly 24%. Using the new hash grid instead of the octree is faster, but still adds an overhead of 9%. The impact on rendering time is evaluated in more detail in Section IV.4.4.

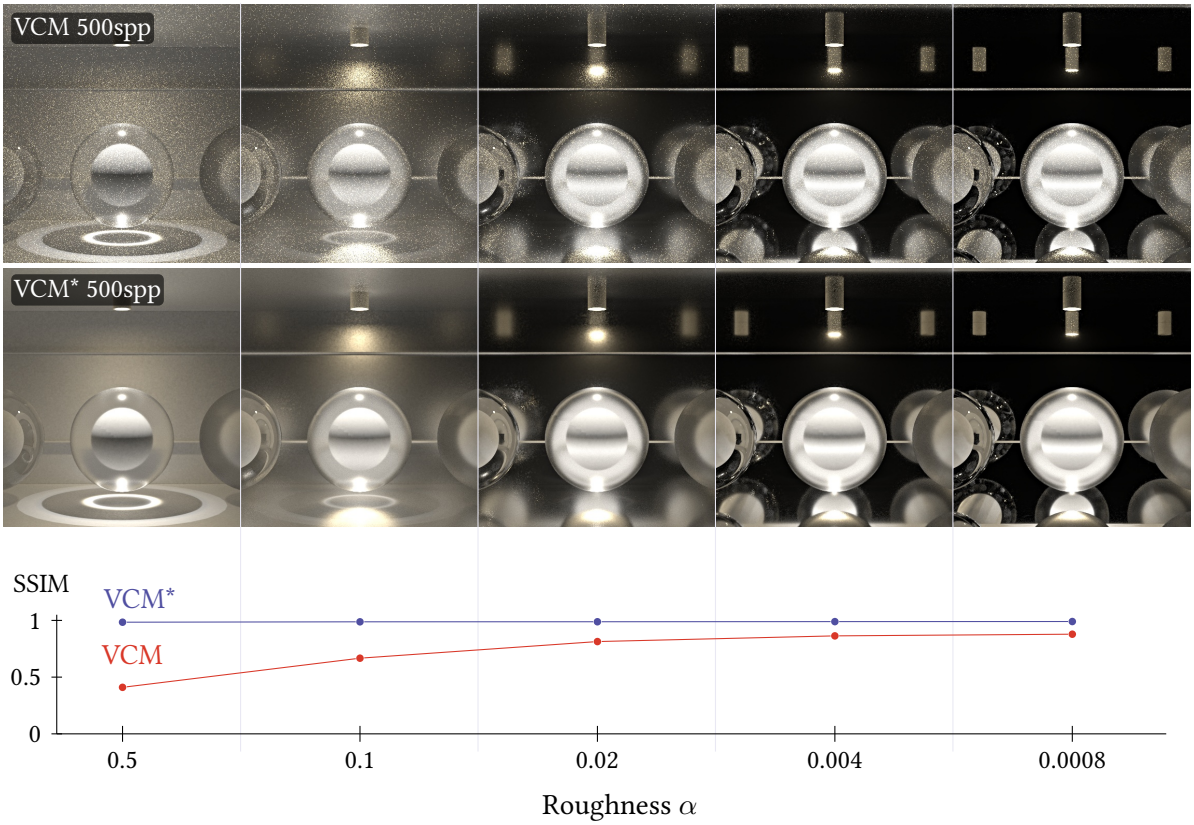


Figure IV.2: Robustness of VCM* over varying BSDFs taken from [Jendersie and Grosch 2018]. A structured similarity (SSIM) [Wang et al. 2004] of 1 matches the reference whereas smaller numbers indicate larger errors. The negative impact of the merge overestimation in VCM (top row) applies even for very smooth surfaces.

IV.2.4 Results

Both heuristics from this section solve the problem of a severe overestimation of the reuse factor as shown in Figure IV.3. Merges closer to the observer (with respect to path vertices) get larger MIS weights than those close to a light source. However, N_R might be estimated too small, because the density of photons from all light sources and path lengths are summed in the density structure. This and the subtle differences between both techniques are better visible in the variance comparison in Section IV.4.

Figure IV.2 additionally demonstrates the robustness for different material setups. The smoother a material, the less scattering it introduces. This decreases the variance of the view sub-path and should reduce the discrepancy between N_R and N_Φ. However, even for fairly smooth surfaces ($\alpha = 0.004$) the effect is still directly visible on the rear wall.

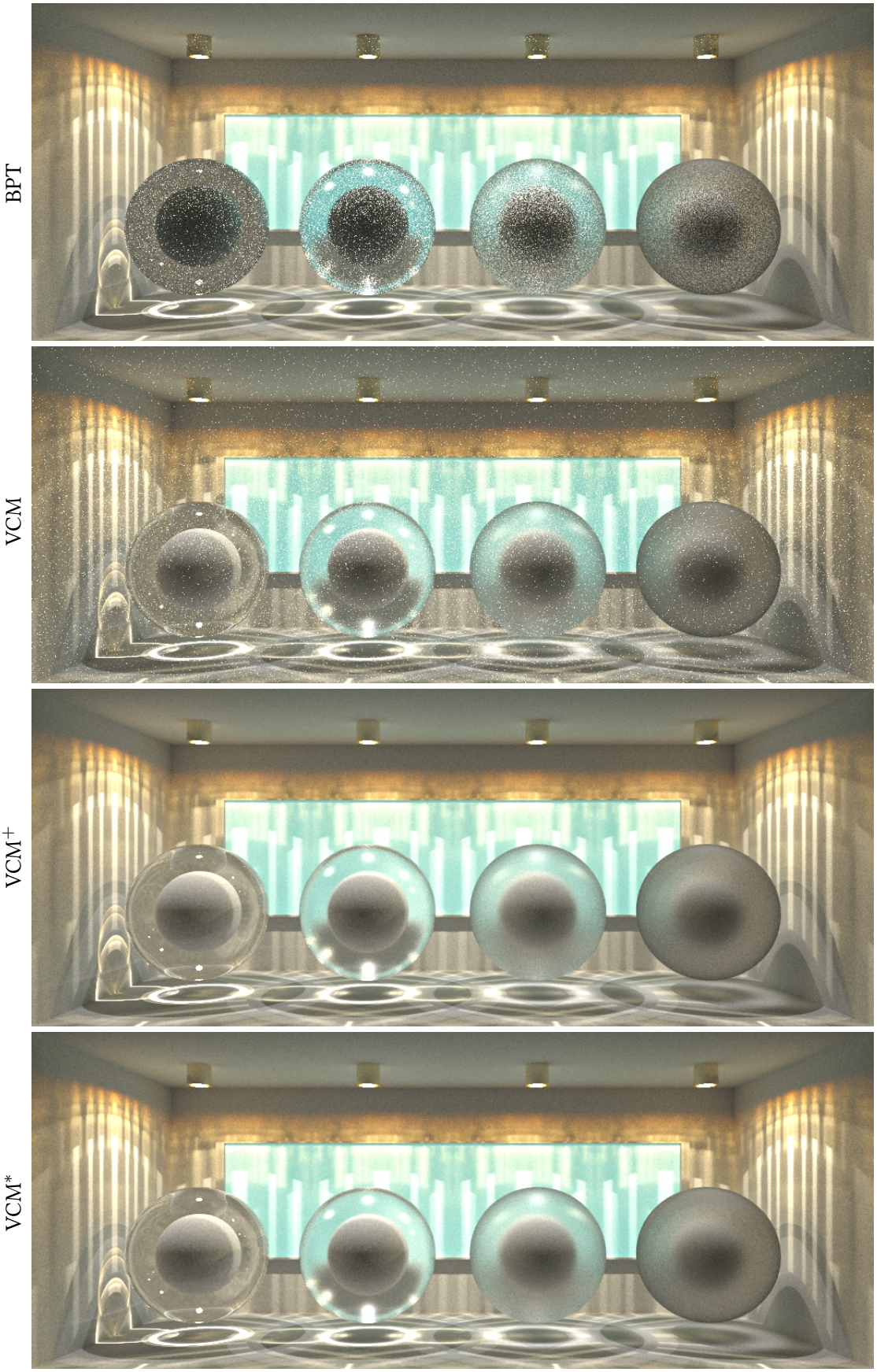


Figure IV.3: Improved merge weighting in VCM$^+$ and VCM* (at 256 spp). The scene behaves similar to Veach's scene, as it has reflecting surfaces close to the light source. Beyond that, it shows glossy-diffuse-glossy paths with varying roughness to demonstrate the robustness of the new approach. The caustics are generated by faceted cylinders around the light sources.

IV.3 Path Reuse Variance as a Consequence of Sub-Path Variance

The previous section introduced the correlation of paths as an explanation for the reduced reuse-efficiency. However, we can ask for the change in the variance much more directly [Jendersie 2019b]. If the variance of the view path dominates the entire sampler variance, then reusing the light sub-path sampler will not improve the variance much.

This can be expressed using the product rule for the variance of two random variables

$$\mathrm{V}[XY] = \mathrm{E}[X]^2\,\mathrm{V}[Y] + \mathrm{V}[X]\,\mathrm{E}[Y]^2 + \mathrm{V}[X]\,\mathrm{V}[Y]\,. \qquad \text{V}[XY] \text{ Eq. (II.16c) p. 20}$$

Whenever the variance of one sampler is reduced, only two of the three terms are changed. The third term remains constant, but if exactly this term dominates the entire sum, the total variance reduction will be small. Let V_1 be the variance without reuse and V_{N_Φ} the variance with reuse of the light sub-paths. The total reuse factor is simply the ratio

$$N_R = \frac{\mathrm{V}_1}{\mathrm{V}_{N_\Phi}} \tag{IV.6}$$

between this two variances.

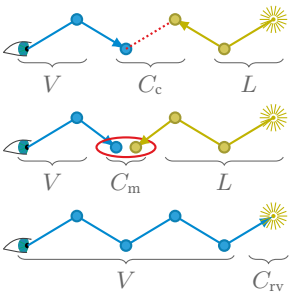
For the sampler types *connections* (t_c), *merges* (t_m) and *random hit* (t_rv) from Section II.7 we can group the terms as follows. Let $V = \prod \frac{\wp}{p}$ be the sampling terms of the view sub-paths, $L = \prod \frac{\wp}{p}$ be the sampling terms from the light sub-path and C be the remaining terms, which can be any of

t_c Eq. (II.79) p. 67
t_m Eq. (II.81) p. 68
t_rv Eq. (II.75) p. 65

$$\text{Connection:}\quad C_\mathrm{c} = \frac{\wp_k \cdot \left|\cos^{\rightarrow}_k\right| \cdot V(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) \cdot \left|\cos^{\leftarrow}_{k+1}\right| \cdot \wp_{k+1}}{\left\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\right\|^2} \tag{IV.7a}$$

$$\text{Merge:}\quad C_\mathrm{m} = \wp_k \cdot K(\boldsymbol{x}_k, \boldsymbol{x}_{k'}) \tag{IV.7b}$$

$$\text{Random Hit:}\quad C_\mathrm{rv} = L_e(\boldsymbol{x}_\ell, \boldsymbol{d}^{\leftarrow}_\ell). \tag{IV.7c}$$



Then,

$$N_R = \frac{\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \mathrm{E}[V]^2\,\mathrm{V}[CL] + \mathrm{V}[V]\,\mathrm{V}[CL]}{\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \frac{1}{N_\Phi}\left(\mathrm{E}[V]^2\,\mathrm{V}[CL] + \mathrm{V}[V]\,\mathrm{V}[CL]\right)} \tag{IV.8}$$

is our searched value. A derivation of Equation (IV.8) can be found in the Appendix A.3.3. Interestingly, the variance of the terms grouped in C also reduces with the reuse of the light sub-paths. This is logical, since for each other light path end vertex, we get another estimate of the BSDF, the kernel or similar.

IV.3.1 Approximate Solution

Unfortunately, we can not directly compute N_R again, because we do not know the variance and expected values. To simplify, we set $\mathrm{V}[C] = 0$. Even though this is a bad assumption (it will practically never be met) the final results are still good. The correct variance of C depends on values like the BSDF and more as visible in Equation (IV.7). However, this trick simplifies Equation (IV.8) to the symmetric form

$$N_R \approx \frac{\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \mathrm{E}[VC]^2\,\mathrm{V}[L] + \mathrm{V}[V]\,\mathrm{E}[C]^2\,\mathrm{V}[L]}{\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \frac{1}{N_\Phi}\left(\mathrm{E}[VC]^2\,\mathrm{V}[L] + \mathrm{V}[V]\,\mathrm{E}[C]^2\,\mathrm{V}[L]\right)} \tag{IV.9}$$

Semantically, the value $\mathrm{E}[CL]$ is the outgoing radiance into the direction of the view sub-path. Analogously, $\mathrm{E}[VC]$ is the outgoing importance into the direction of the light sub-path. I.e. the two values are the expected number of sub-path end vertices per square meter and steradian. In any case, $\mathrm{E}[C]$ can be factored into $\mathrm{E}[C']\,\mathrm{E}[\wp_k]$, such that $\mathrm{E}[\wp_k]$ can be canceled from the above equation (simply imaging Eq. (IV.9) with C' everywhere). Without the BSDF, $\mathrm{E}[C'L]$ is the incoming irradiance E and $\mathrm{E}[VC']$ is the incoming importance, both measured per square meter.

Radiance Sec. II.1.4 p. 17

Irradiance Sec. II.1.3 p. 16

For the variances terms $\mathrm{V}[V]$ and $\mathrm{V}[L]$ we can assume that they are close to zero as long as we have sufficient importance samplers:

$$\mathrm{V}\left[\prod_i \frac{\wp_i}{p_i}\right] = \varepsilon \approx 0$$

This assumption holds for many cases:

- Specular events have $\mathrm{V} = 0$ (deterministic)
- The Lambertian diffuse BRDF (Sec. II.5.3 p. 44) has a perfect importance sampler
- A good sampler for the Oren-Nayar model is derived in Sec. II.5.4 p. 45
- Microfacet models (Sec. II.5.5 p. 49) have a good sampler [Heitz and d'Eon 2014]
- Mixed materials of the above simpler materials can be sampled equally well through multiple importance sampling

MIS II.2.3 p. 22

Hence, we can simplify the equation for the reuse count further to

$$\begin{aligned} N_R &\approx \frac{\varepsilon\mathrm{E}[C'L]^2 + \varepsilon\mathrm{E}[VC']^2 + \varepsilon^2\mathrm{E}[C']^2}{\varepsilon\mathrm{E}[C'L]^2 + \frac{1}{N_\Phi}\left(\varepsilon\mathrm{E}[VC']^2 + \varepsilon^2\mathrm{E}[C']^2\right)} \\ &\approx \frac{\mathrm{E}[C'L]^2 + \mathrm{E}[VC']^2}{\mathrm{E}[C'L]^2 + \frac{1}{N_\Phi}\mathrm{E}[VC']^2} \end{aligned} \tag{IV.10}$$

where $\varepsilon^2\mathrm{E}[C']^2$ can be removed, because it is much closer to zero than the other two terms due to ε squared. The formula fulfills all requirements $1 \leq N_R \leq N_\Phi$ without additional modifications.

The two required expected values are the densities per square meter of the respective sub-path sampler. For both we would need to compute an integral which is infeasible at this point. Instead we want to predict the densities, using the sampled path only. Let A_X be the *footprint area* of a

sampler X and $P(X) \in [0,1]$ be the product of discrete probabilities from Russian roulette decisions. For example, if there are multiple light sources $P(L) < 1$ is the probability to start with the specific light source. Then, the two expected values can be expressed as:

$$E[C'L] = \frac{P(L)}{A_L} \qquad \text{and} \qquad E[C'V] = \frac{P(V)}{A_V} \tag{IV.11}$$

where the estimation of A_X is the topic of the next section.

IV.3.2 Sample Footprints

Let the *footprint* $\mathcal{A}[X]$ be the set of all positions which can be reached through the sampler X. Then, $p_{\mathcal{A}[X]} : \mathcal{A}[X] \mapsto \mathbb{R}^+$ is the probability density for all these positions which is normalized over the entire scene surface $\mathcal{M}$:

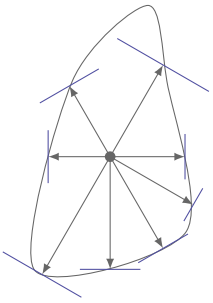
$$\int_{\mathcal{M}} p_{\mathcal{A}[X]}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1.$$

The *footprint area* of sampler X at position $\boldsymbol{x}$ is the reciprocal value

$$A_X(\boldsymbol{x}) = \frac{1}{p_{\mathcal{A}[X]}(\boldsymbol{x})}. \tag{IV.12}$$

Assuming a Gaussian model for $p_{\mathcal{A}[X]}$, centered around $\boldsymbol{\mu} = \boldsymbol{x}$ with a covariance matrix $\boldsymbol{\Sigma}$, we have

$$\begin{aligned} A_X(\boldsymbol{x}) &= \frac{1}{\frac{1}{2\pi\sqrt{|\boldsymbol{\Sigma}|}} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))} \\ &= \frac{1}{\frac{1}{2\pi\sqrt{|\boldsymbol{\Sigma}|}} \exp(-\frac{1}{2}\mathbf{0}^T\boldsymbol{\Sigma}^{-1}\mathbf{0})} = \frac{1}{\frac{1}{2\pi\sqrt{|\boldsymbol{\Sigma}|}} \exp(-0)} \\ &= 2\pi\sqrt{|\boldsymbol{\Sigma}|} \end{aligned} \tag{IV.13}$$



Footprint areas

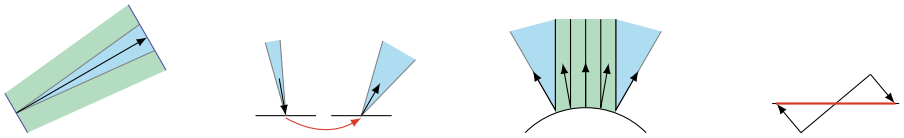
which means that the *footprint area* is connected to the variance of distances around $\boldsymbol{x}$.

Another perspective is that if we distribute N samples with sampler X which end in positions $\boldsymbol{x}_i$, $A_X(\boldsymbol{x}_i)$ describes the expected size of a Voronoi region around each sample $\boldsymbol{x}_i$.

The first footprint estimate for Ray Tracing was introduced by Igehy [1999]. His *ray differentials* estimate the derivatives of pixel positions with respect to the final surface of the view sub-path. From this derivatives an anisotropic area can be calculated which is then applied to filtering of texture samples. However, *ray differentials* are only able to model specular interactions with the scene correctly.

Suykens and Willems [2001] introduced a treatment of BSDFs to Igehy's *ray differentials*, calling them *path differentials*. Therefore, it is necessary to provide the partial derivatives for each used sampling routine of the materials. However, their approach requires an arbitrary scale parameter that is hard to determine. They applied the *path differentials* to texture antialiasing too and as refinement oracle for hierarchical radiosity. In some way, *path*

Table IV.1: Comparison of simple Gaussian model-based footprint estimates. Notation: a = square root of footprint area A, υ = square root of solid angle ω, d = travel distance, p = sampling PDF of a material, $\sigma_{\boldsymbol{x}}^2$ = positional variance of samples, $\sigma_\triangleleft$ angular change of positional variance, H = mean curvature, h = parameter.



Method	Transfer	Material	Curvature	Projection
OVCM [J19]	$a' = a + \upsilon \cdot d$	$\upsilon' = \upsilon + 1/\sqrt{p}$	No	No
[JG19]	$\sigma'_{\boldsymbol{x}} = \sigma_{\boldsymbol{x}} + \sigma_\triangleleft \cdot d$	$\sigma'_\triangleleft = \sigma_\triangleleft + 1/\sqrt{2\pi p}$	$\sigma'_\triangleleft = \sigma_\triangleleft + \sigma_{\boldsymbol{x}} \cdot \lvert H\rvert$	No
[BSCHS03]	$a' = a + \upsilon \cdot d$	$\upsilon' = h/\sqrt{p}$	No	$a' = a/\sqrt{\cos\theta}$
IVCM (Thesis)	$\sigma_{\boldsymbol{x}}^{2\prime} = \sigma_{\boldsymbol{x}}^2 + \sigma_\triangleleft^2 \cdot d^2$	$\sigma'_\triangleleft = \sigma_\triangleleft + 1/\sqrt{2\pi p}$	$\sigma'_\triangleleft = \sigma_\triangleleft + \sigma_{\boldsymbol{x}} \cdot \lvert H\rvert$	$\sigma_{\boldsymbol{x}}^{2\prime} = \sigma_{\boldsymbol{x}}^2/\lvert\cos\theta\rvert$

differentials were used in RenderMan too [P. H. Christensen et al. 2003], using a simplified version. In RenderMan, the derivatives of glossy reflections were set to the solid angle of the sampled cone divided by the number of rays used for this glossy reflection.

A different application of *ray differentials* is the estimation of photon footprints for adaptive photon mapping. *Photon differentials* [Schjøth et al. 2007] apply *ray differentials* to photons. After choosing two initial offset vectors at the light source, Igehy's *ray differentials* are applied for further specular interactions. Only specular photon paths are stored in the caustic map and a treatment of BSDFs is not required or given.

The most convenient solution so far is the *5D Covariance Tracing* from Belcour et al. [2013]. It makes use of the Gaussian model noted above and models many interactions including a proper handling of BSDFs and occlusion. The five dimensions are two spatial, two angular and one temporal component. It requires 15 values to store the symmetric 5D matrix while tracing.

Over the course of publications, I used different estimates similar to the covariance tracing. These are simplifications which reduce the memory and computational overhead compared to the 5D approach. Table IV.1 summarizes the estimates in a unified manner. In a work of Bekaert et al. [2003] another simplified version of footprints was used, which is also shown in the table.

IV.3.3 Footprint Approximation

In this section the details of the approximations in Table IV.1 are explained. As mentioned above, the footprint is estimated as a Gaussian distribution. More precisely, an isotropic 2D kernel is assumed such that the two spatial standard deviation values are the same and only one value ($\sigma_{\boldsymbol{x}}$) is computed and stored. Otherwise twice as many values and additional overhead for rotation operators would be necessary.

According to Equation (IV.13) the area is then reconstructed by

$$A = 2\pi\sigma_{\boldsymbol{x}}^2.$$

Contrarily, we use the inverse transformation $\sigma_{\boldsymbol{x}} = 1/\sqrt{2\pi p}$ to transform a per area density p into a standard deviation parameter.

Additionally, we define $\sigma_\triangleleft = \mathrm{d}\sigma_{\boldsymbol{x}}/\,\mathrm{d}d$ as the rate of change of $\sigma_{\boldsymbol{x}}$ over the travel distance d. After using different models, this formulation proved to be the best in my experiments. It is surely not the only way to define a footprint estimate.

Since the constant 2π appears in every term, we can neglect it when converting back and forth. Indeed, the parameters of the two other estimates in Table IV.1 are connected, namely by $a = \sigma_{\boldsymbol{x}}\sqrt{2\pi}$ and $\upsilon = \sigma_\triangleleft\sqrt{2\pi}$. They have a direct geometric interpretation. a is the square root of the area $\sqrt{A}$ and υ is the square root of a solid angle $\sqrt{\omega}$. Due to this connection I will use υ and a if focusing on geometric interpretations and $\sigma_{\boldsymbol{x}}$ and $\sigma_\triangleleft$ otherwise. It is always possible to convert between these two by applying the constant $\sqrt{2\pi}$.

Initialization

Dependent on the source type of the current path, we need to set different values for the parameters. Most terms are straight forward, since the sampling densities are well known. If $p_{\boldsymbol{x}}$ is the per area density to sample the first vertex of the path and $p_\triangleleft$ is the sampling density of the first direction, then

$$\sigma_{\boldsymbol{x}}^2 = \frac{1}{2\pi p_{\boldsymbol{x}}} \qquad \text{and} \qquad \sigma_\triangleleft = \frac{1}{\sqrt{2\pi p_\triangleleft}}$$

Table IV.2 summarizes the resulting terms for the most common sources. A point light source has zero extent and distributes the samples uniformly over the entire sphere, so $\sigma_\triangleleft$ is a constant. Uniform area lights have a known source area and directions are sampled according to a cosine distribution. For orthogonal lights the source area is that of the projected bounding box if sampling starting positions on this boundary. Only environment lights must be treated very carefully. The parameter $\sigma_\triangleleft$ must be set such that $\Delta\sigma_{\boldsymbol{x}}^2 = {}^1\!/_{2\pi p_\triangleleft}$ after the first transfer operation. Thereby $p_\triangleleft$ is the angular PDF to sample the selected direction in the environment map. Due to orthogonal transfer there is no scaling with the distance, or the first distance must be included in the initialization, which is done in the table.

Table IV.2: Initialization values for the footprint estimate.

	a	υ	$\sigma_{\boldsymbol{x}^2}$	$\sigma_\triangleleft$
Point Light (Sec. II.3.1 p. 29)	0	$\frac{1}{\sqrt{4\pi}}$	0	$\frac{1}{\sqrt{8\pi}}$
Area Light (Sec. II.3.2 p. 30)	$\sqrt{A_{\text{source}}}$	$\sqrt{\pi/\cos\theta}$	$\frac{A_{\text{source}}}{2\pi}$	$\frac{1}{\sqrt{2\cos\theta}}$
Directional Light (Sec. II.3.3 p. 32)	$\sqrt{A_{\text{BB}_\perp}}$	0	$\frac{A_{\text{BB}_\perp}}{2\pi}$	0
Environment Light (Sec. II.3.4 p. 33)	$\sqrt{A_{\text{BB}_\perp}}$	$\frac{1}{d\sqrt{p_\triangleleft}}$	$\frac{A_{\text{BB}_\perp}}{2\pi}$	$\frac{1}{d\sqrt{2\pi p_\triangleleft}}$
Pinhole Camera (Sec. II.6.1 p. 59)	0	$\frac{1}{\sqrt{p_{\text{Cam}}}}$	0	$\frac{1}{\sqrt{2\pi p_{\text{Cam}}}}$

p_{Cam} Eq. (II.63) p. 60

Free Space Transfer Operator

All methods in Table IV.1 have the transfer operator trough free space in common. From Section II.7.3 we know how to transform a per solid angle PDF into a per area PDF at a target region. While traveling, the area grows

Conversion of PDF at end of Sec. II.7.3 p. 65

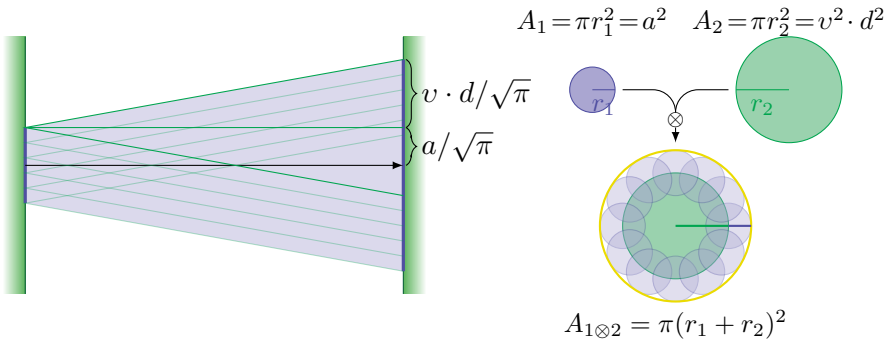
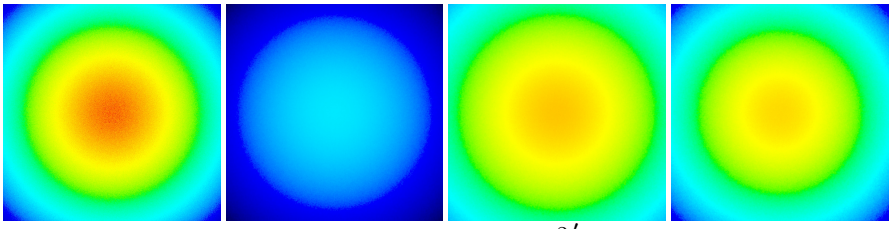
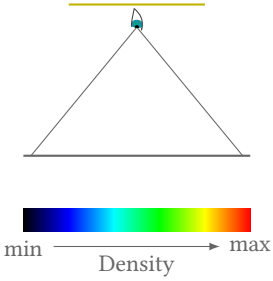


Figure IV.4: Geometrical model of the transfer operator. The convolution of the area from angular scattering and the source area is a summation of radii. As shown in this section, this model is not as good as the Gaussian distribution model.



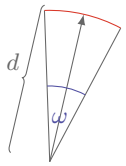
(a) LT (b) $\sigma_{\boldsymbol{x}}{}' = \sigma_{\boldsymbol{x}} + \sigma_{\triangleleft} \cdot d$ (c) $\sigma_{\boldsymbol{x}}^{2\prime} = \sigma_{\boldsymbol{x}}^2 + \sigma_{\triangleleft}^2 \cdot d^2$ (d) (c)+Projection



Here and in the following, the values of min and max are selected manually.

Figure IV.5: Empirical study of the transfer operator. The density received on a planar surface, under illumination of an area light, is shown. Adding variances (c and d) comes closer to the correct result than adding standard deviations (b).

quadratically with the distance ($A = \omega d^2$). Working with square roots, v must be multiplied with the travel distance d: $a = vd$. Geometrically $\omega \cdot d^2$ is the area of the spherical cap after a distance of d and $v \cdot d$ is simply the square root of this term.



In addition to the angular scattering, there is also the source area. Assuming a locally constant material, each point scatters the particles accordingly. Hence, we have to use a convolution of the two areas at the target. In the ray tracing gem [Jendersie 2019b] I used a geometrical interpretation of this convolution which is shown in Figure IV.4. However, using the Gaussian distribution model, we obtain a different solution for the convolution. Given the covariance matrices Σ_1 and Σ_2 of two zero centered Gaussians, the covariance matrix of their convolution is ([Bromiley 2003])

$$\Sigma_{1\otimes 2} = \Sigma_1 + \Sigma_2. \tag{IV.14}$$

This means we have to sum up variances and not standard deviations which results in

$$\sigma_{\boldsymbol{x}}^{2\prime} = \sigma_{\boldsymbol{x}}^2 + \sigma_{\triangleleft}^2 \cdot d^2. \tag{IV.15}$$

To check whether the previous geometrical construction or the Gaussian perspective is correct, I conducted the following empirical experiment. Figure IV.5 visualizes the photon density on a planar receiver, which is illuminated from a quadratic, parallel aligned area light. The ground truth is created with a modified light tracer, which ignores the BSDF of the last

Light tracing Sec. II.8.4 p. 74

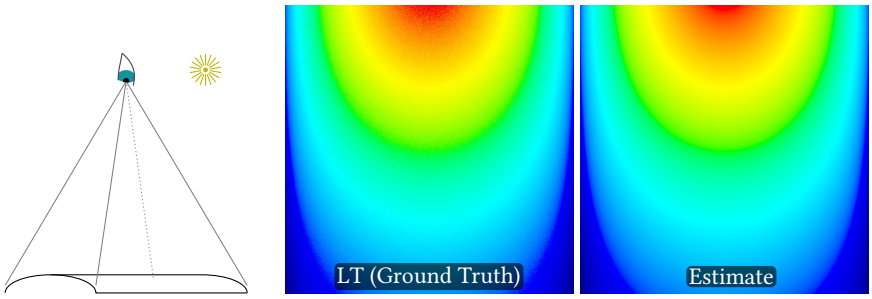


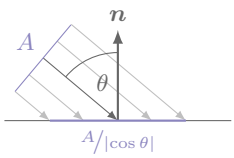
Figure IV.6: Validation of cosine projection $\sigma_{\boldsymbol{x}}^{2\prime} = \sigma_{\boldsymbol{x}}^2 / |\cos\theta|$. The scene consists of a cylindrical surface which is illuminated by a point light source. The footprint estimate perfectly matches the ground truth.

surface and the throughput value. With this change, it shows the incident particle density instead of radiance.

To visualize the density from the footprint estimate, the nearest particle on the surface is searched and its value $1/A$ is directly displayed. After a single iteration this yields constant valued Voronoi regions around all particles. Integrating over time averages the density estimates from particles of different paths. In the ideal case, if the estimate is correct, any particle at a position would have the same value, which is not the case.

Concluding this experiment, adding the variance, as required by the Gaussian model, is closer to the ground truth. If we add projections as explained in the next section, the result will be even better. The underestimated density in the center region is due to the Gaussian assumption. Each point on the quadratic light source with sharp boundaries is handled as if the other points form a symmetric Gaussian, which is clearly not correct.

Projection



Lambert [1760] observed that the particle density on a surface decreases with the cosine of the incident direction. Therefore, it is necessary to scale the area with the reciprocal cosine. The flatter the incident direction, the smaller the density and the larger the footprint area.

When leaving a surface, the inverse transformation must be applied. In that case we need to multiply with the respective cosine value.

In previous publications I left out the projection on purpose. Experimentally, the desired outcome was better without applying the projection than with it. This changed with the different handling of convolutions in the previous section. When adding variances instead of standard deviations, the projection works out as expected. Besides the result in Figure IV.5 (d), Figure IV.6 shows an additional validation experiment.

Material Scattering

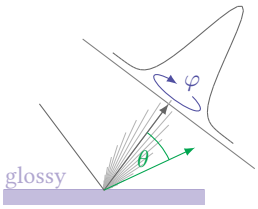
The scattering of particles by surface interactions is the most important part of the proposed footprint estimate. Materials are the reason why we cannot apply ray differentials [Igehy 1999] as an estimator for the expected value.

The goal is to find a rate of density change σ_p which depends on the BSDF such that

$$\sigma_\triangleleft{}' = \sigma_\triangleleft + \sigma_p.$$

To determine σ_p we can look at the desired outcome. We know that the sampling PDF $p(\boldsymbol{d}_\uparrow)$ of the BSDF describes the angular density of outgoing directions. This can be converted into a per area density by dividing with the squared travel distance. After applying the transfer operator, the resulting Gaussian density must have the same value at the origin:

$$\frac{p(\boldsymbol{d}_\uparrow)}{d^2} = g\left(\boldsymbol{x} = \mathbf{0} \,\middle|\, \Sigma = \begin{bmatrix} \sigma_p^2 d^2 & 0 \\ 0 & \sigma_p^2 d^2 \end{bmatrix}\right) = \frac{1}{2\pi\sigma_p^2 d^2}$$
$$\Leftrightarrow \frac{1}{\sqrt{2\pi p(\boldsymbol{d}_\uparrow)}} = \sigma_p \tag{IV.16}$$



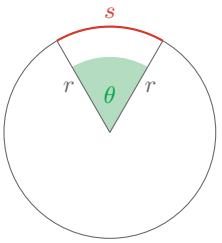
It is possible to interpret σ_p as the standard deviation of a Gaussian in the tangential plane at distance one. This perspective was used in my previous publication [Jendersie and Grosch 2019] and produces the same result.

Further, the derivation of σ_p is also used for the initialization of $\sigma_\triangleleft$ in table IV.2. In any case, the density after applying the transfer operator must match the known single segment per area density of the sampling events.

Curvature

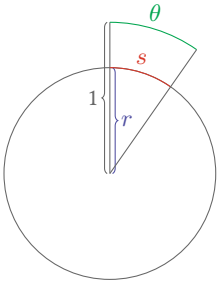
The last term I used for footprint estimates is based on curvature. If the BSDF is a scattering term on microscopic level, then curvature leads to a similar scattering on macroscopic level. In the regularization paper [Jendersie and Grosch 2019] I introduced curvature in exactly this way. By using an absolute term $|H \cdot \sigma_{\boldsymbol{x}}|$ for $\sigma_\triangleleft$, where H is the mean curvature, any bent surface will increase the angular scattering.

The following ideas are highly experimental and should be used with care. I tried to verify the derived formulas experimentally, but most experiments produced (sometimes significant) differences to the ground truth. Hence, the proposed approximations are either too coarse or there might be errors in the derivation.



A curvature value κ is defined as the derivative of the angle between two normals over the arc length s of a curve: $\mathrm{d}\theta/\,\mathrm{d}s$. H is simply the average curvature on a 2D manifold. For a sphere $\kappa = H$ is isotropic and equals $1/r$ for its radius r. Now, using the intercept theorem we get a geometrical interpretation of the term H times a length s. With $1 : r = \theta : s$ we obtain an angle

$$\theta = \frac{1}{r} \cdot s = H \cdot s \tag{IV.17}$$



which is the change of the normal over a tangential deviation of s.

In the previous publication I falsely used $|H \cdot \sigma_{\boldsymbol{x}}|$ for the rate of change $\sigma_\triangleleft$ itself. However, we are actually not interested in the change of the normal Hs, but in the change of the scattered direction ω which depends on the reflection or refraction operator. This will be discussed in the following.

Replacing s with $\sigma_{\boldsymbol{x}}$ is no problem. While $\sigma_{\boldsymbol{x}}$ is not a length s, but a standard deviation, the two are connected by a constant factor

$$s = \sqrt{\frac{A}{\pi}} = \sqrt{\frac{2\pi\sigma_{\boldsymbol{x}}^2}{\pi}} = \sqrt{2}\sigma_{\boldsymbol{x}}$$

when using the introduced connection to the area. Once we obtain the desired scattering angle ω, it must be converted back into a rate of standard deviation $\sigma_\triangleleft$. This removes the $\sqrt{2}$ factor again.

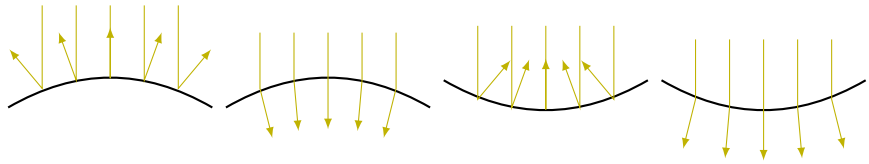
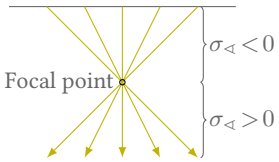


Figure IV.7: Influence of convex and concave surfaces onto footprint convergence or divergence.

Further, I used the absolute values, because the signed values produced bad results which will also be demonstrated in the following. It still makes sense that scattering of the footprint increases with curvature.

Nevertheless, using the absolute value $|H \cdot \sigma_{\boldsymbol{x}}|$ is clearly not correct. Instead, it is also possible that the scattering will be reduced or invert its direction. For example a convex lens will focus a ray bundle, causing a focal point with zero area in a certain distance. For reflections the sign only depends on the sign of the curvature with respect to the incident direction. A convex surface like a mirroring ball will increase the scattering, while a concave mirror will decrease it (see Figure IV.7). For refractions the amount of scattering also depends on the ratio of refraction indices η_i/η_t.

Refraction Sec. II.5.2 p. 40



Using a negative rate $\sigma_\triangleleft < 0$ itself is sound, but the Gaussian model does not allow a negative valued standard deviation $\sigma_{\boldsymbol{x}}$ or variance in the outcome. This problem can be solved by taking the absolute value of both $\sigma_{\boldsymbol{x}}^2$ and $\sigma_\triangleleft$ if $\sigma_{\boldsymbol{x}}^2 < 0$ after applying the travel operator. Geometrically the switch of the sign happens when passing the focal point. Beginning at the focal point, the rays will diverge with the same rate as they converged before the point and the area in any distance will be positive again.

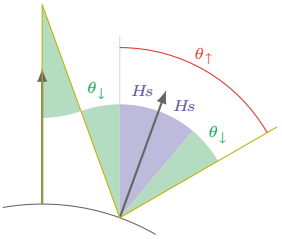
In the following I will use

$$H_\downarrow = H \operatorname{sign}(\cos\theta_\downarrow) \tag{IV.18}$$

to denote the signed curvature as seen from the incident direction. The computation of H is detailed in Appendix A.4.2.

Reflection

As the *law of reflection* states that the angle of incidence $\theta_\downarrow$ equals the exitant angle $\theta_\uparrow$. So, if $H_\downarrow s$ is the change of the normal angle, the outgoing scattering will be twice as large:



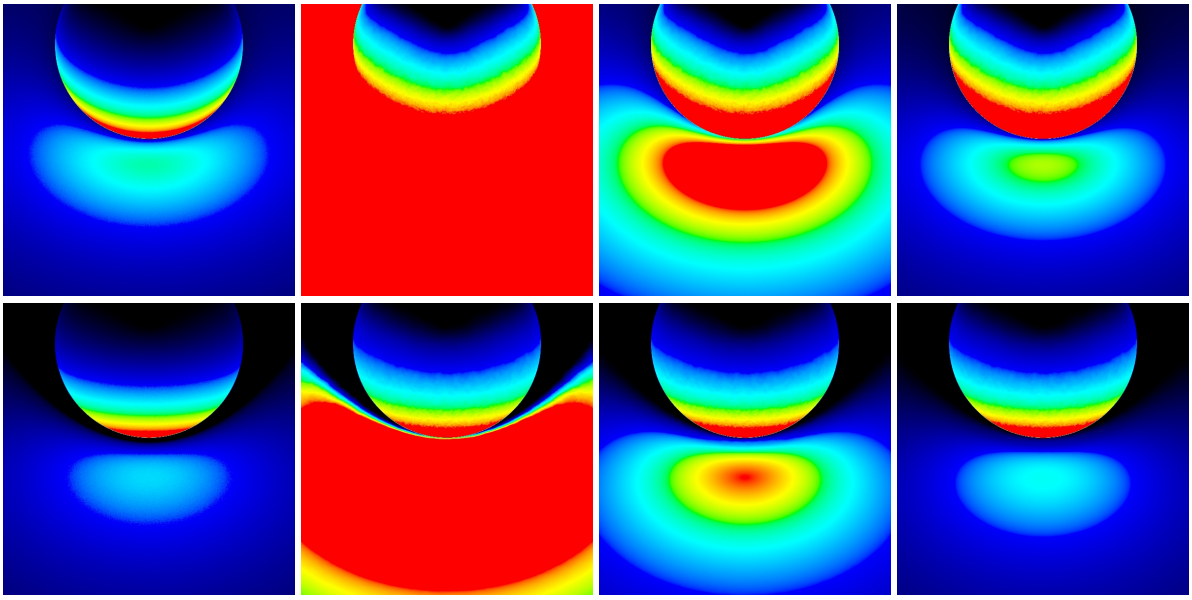
$$\theta_\uparrow = \theta_\downarrow + 2H_\downarrow s \tag{IV.19}$$

$$\Rightarrow \quad \sigma'_\triangleleft = \sigma_\triangleleft + 2H_\downarrow \sigma_{\boldsymbol{x}}. \tag{IV.20}$$

When squaring $\sigma'_\triangleleft$ for the transfer operation, the sign must be kept.

In Figure IV.8 the reflection of a mirror ball onto a planar floor is shown. The density on the mirror ball itself is from the diffuse scattered light on the floor. The overestimation is an artifact from short travel distances. Since the source area of particles is larger than the travel distance, the assumptions in the travel operator are violated. It assumes that all particles on the source area have the same travel distance d to the target, which is not the case.

The different shape in the reflection can be explained by a violation of the isotropy assumption. The projected footprint on the sphere is not a circle, which is assumed in any operator of the proposed estimate.



(a) LT (Ground Truth) (b) No curvature (c) $|H\sigma_{\boldsymbol{x}}|$ (d) Eq. (IV.20)

Figure IV.8: Convex reflections from a mirror ball onto a planar receiver. In the top row an orthographic light source is used and in the bottom row a point light source.

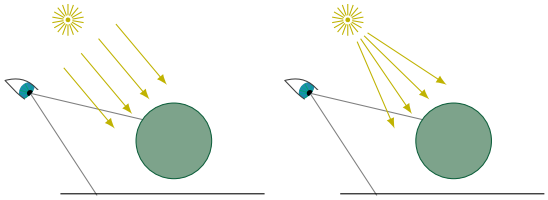


Figure IV.9: Schematic overview of Figure IV.8.

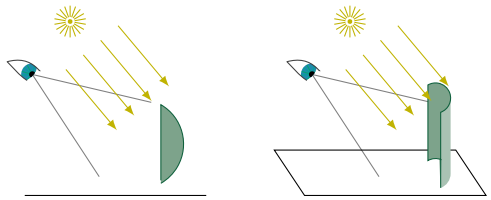
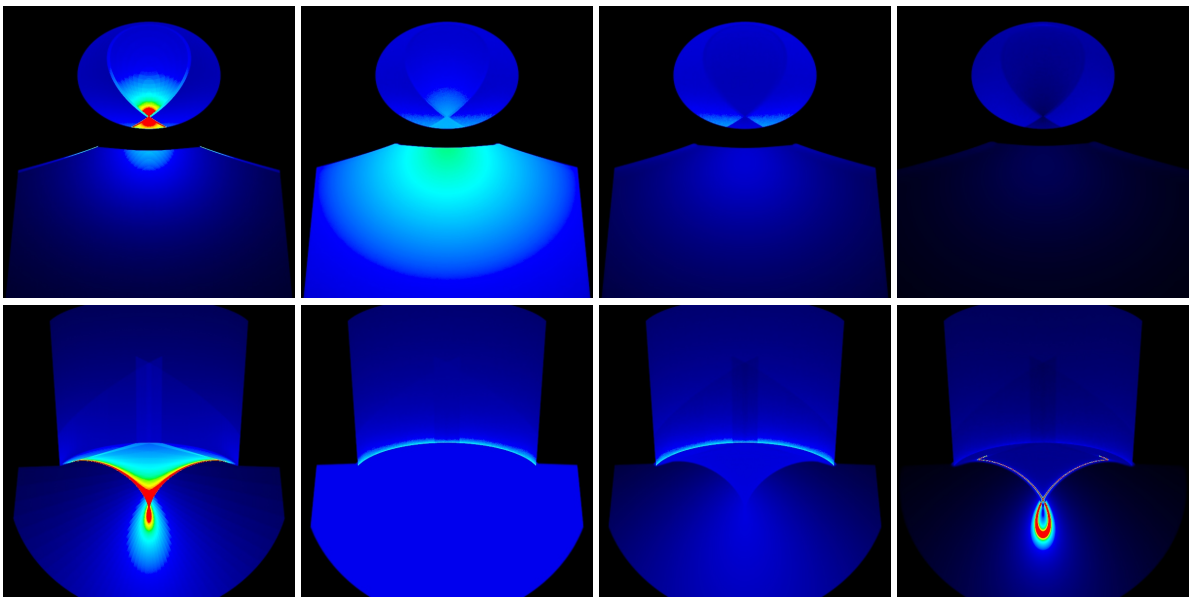


Figure IV.10: Schematic overview of Figure IV.11.



(a) LT (Ground Truth) (b) No curvature (c) $|H\sigma_{\boldsymbol{x}}|$ (d) Eq. (IV.20)

Figure IV.11: Examples for concave reflections. Both scenarios have a planar receiver and are illuminated by an orthographic light source. The upper row shows the density of a spherical cap mirror and the lower row that of a cylindrical mirror.

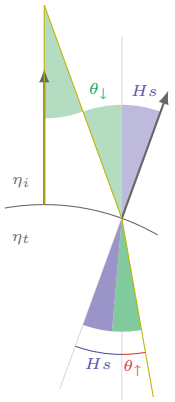
Unfortunately, any experiment with concave surfaces failed, because the approximation using only isotropic kernels is too coarse. Tracing an average radius and using the mean curvature produces focal points of the cone estimate at the wrong positions. Even for a simple scenario, where we have an isotropic curved surface, the angle of incidence causes an anisotropic footprint. Treating things in an isotropic manner then produces a focal point at the wrong distance. For non-isotropic surfaces things become even worse. Figure IV.11 demonstrates two examples.

Refraction

For refraction the incident and exitant angles are connected by Snell's law. Even if there is no curvature, a refraction into a denser medium will reduce the scattering angle and vice versa. For the geometrical angles we get

Snell's law Eq. (II.42) p. 42

$$\arcsin\left(\frac{\eta_i}{\eta_t}\sin(\theta_\downarrow + H_\downarrow s)\right) = H_\downarrow s + \theta_\uparrow$$



as can be seen in the construction on the right. We are interested in the first derivative of

$$\theta_\uparrow(s) = \arcsin\left(\frac{\eta_i}{\eta_t}\sin(\theta_\downarrow + H_\downarrow s)\right) - H_\downarrow s \tag{IV.21}$$

at $s = 0$ to describe the change of the scattering angle $\theta_\uparrow$ over the arc length s. This is the first order Taylor approximation which produces

$$\begin{aligned}\frac{\partial\theta_\uparrow}{\partial s}(0) &= \frac{\eta_i}{\eta_t}\frac{\cos(\theta_\downarrow)}{\sqrt{1-\frac{\eta_i}{\eta_t}^2(1-\cos^2(\theta_\downarrow))}}H_\downarrow - H_\downarrow.\\ &= \frac{\eta_i}{\eta_t}\frac{\cos(\theta_\downarrow)}{\cos(\theta_\uparrow)}H_\downarrow - H_\downarrow.\end{aligned} \tag{IV.22}$$

Thus, we can compute the scattering angle under the assumption of parallel incident rays by multiplying the above equation with $\sigma_{\boldsymbol{x}}$. I was able to verify that the above formula produces the correct focus distances for parallel incident rays in a constructed 2D example. However, it misses the incident scattering υ. If adding this term to $\theta_\downarrow$, the result will contain $\cos(\theta_\downarrow + \upsilon/\sqrt{\pi})$ terms. This is unfeasible since $\sigma_\triangleleft$ and therefore any converted υ are unbounded. Taking the cosines of these values simply does not makes sense anymore.

If we assume that the previous scattering is independent of local angles, we can apply Snell's law directly

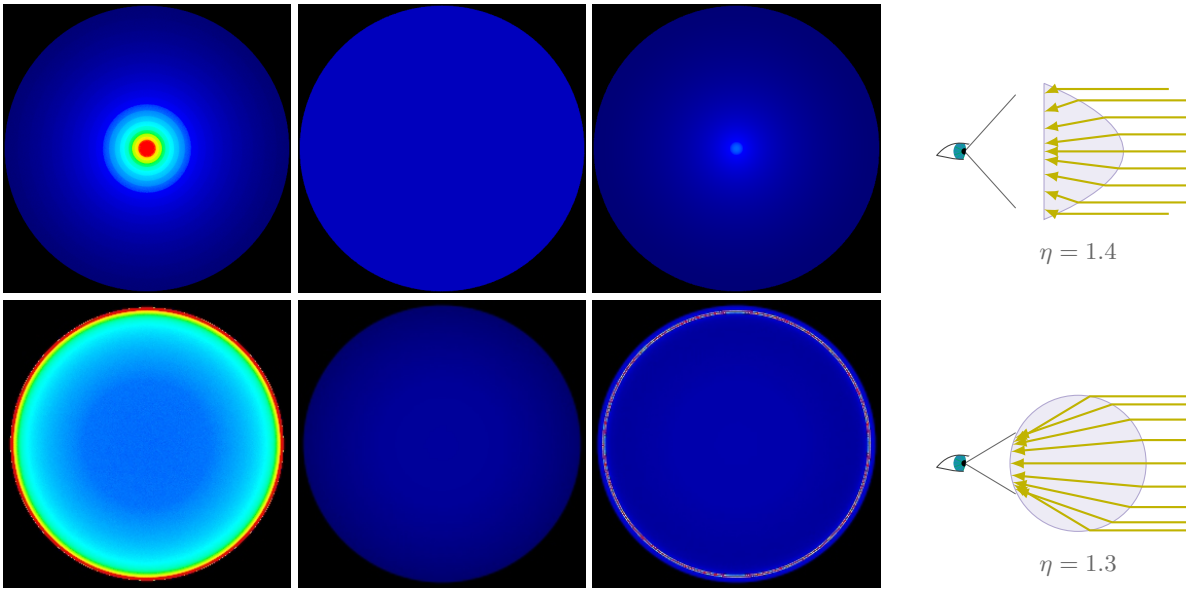
$$\begin{aligned}\upsilon_\uparrow &= \sqrt{\pi}\arcsin\left(\frac{\eta_i}{\eta_t}\sin(\upsilon_\downarrow/\sqrt{\pi})\right)\\ \mathrm{d}\upsilon_\uparrow &= \frac{\eta_i}{\eta_t}\frac{\cos(\upsilon_\downarrow/\sqrt{\pi})}{\cos(\upsilon_\uparrow/\sqrt{\pi})}\,\mathrm{d}\upsilon_\downarrow\\ &= \frac{\eta_i}{\eta_t}\,\mathrm{d}\upsilon_\downarrow\end{aligned}$$

Evaluation at $\upsilon_\downarrow = \upsilon_\uparrow = 0$

meaning that the angular density changes with a factor η_i/η_t

Putting the things together, the update formula for refractions is

$$\sigma'_\triangleleft = \frac{\eta_i}{\eta_t}\sigma_\triangleleft + \left(\frac{\eta_i}{\eta_t}\frac{\cos\theta_\downarrow}{\cos\theta_\uparrow} - 1\right)H_\downarrow\sigma_{\boldsymbol{x}}. \tag{IV.23}$$



(a) LT (Ground Truth) (b) No curvature (c) Eq. (IV.23)

Figure IV.12: Density on the backside of a refractive paraboloid and a refractive sphere. The refraction estimate is closer to the ground truth than using no curvature. However, it is still far away from the expected result.

Figure IV.12 shows the results of the verification experiment. Unfortunately, the derived result is far off the ground truth.

Discussion: Reflection and Refraction Terms

The derived formulas perform poorly for any case where $\sigma_\triangleleft$ becomes negative. The major reason is the incompatibility of the Gaussian density approach and the geometric quantities. Using the geometric transport factor $a' = a + v \cdot d$, I was able to validate some of the paths in the paraboloid and sphere examples. However, the experiment in Figure IV.5 has clearly shown that using the Gaussian convolution approach $\sigma_{\boldsymbol{x}}^{2'} = \sigma_{\boldsymbol{x}}^2 + \sigma_\triangleleft \cdot d^2$ mixes the densities in a more meaningful way.

Additionally, the assumptions (planarity, isotropy and independence of events) can be violated. For rays which hit a surface at an arbitrary angle, the footprint becomes a (deformed) ellipsis. This already deviates from the isotropy assumption.

As a final measure I applied the formulas to the application case of computing the reuse factor. In general, simpler is better. The fewer inputs are required for a similar quality, the faster the computation. When computing MIS weights, all events along a path are evaluated. This means that for each segment of the path, all of the introduced update formulas must be applied. Interestingly, the difference between the signed terms (Equations (IV.20) and (IV.23)) and the simpler

Reuse factor Eq. (IV.10) p. 108

$$\sigma'_\triangleleft = \sigma_\triangleleft + \sigma_{\boldsymbol{x}} \cdot |H| \tag{IV.24}$$

are almost non-existent. In a test over three scenes, the moderate additive scattering with $|H|$ performed slightly better on average than all other variants, including no curvature term at all. Therefore, Equation IV.24 will be used in the following section.

IV.4 Evaluation of Reuse Factor Computations

At the beginning of this chapter, the problem of overestimated reuse factors in VCM was introduced. The true reuse factor in MIS weight computations is smaller than the total number of photons, since only a part of the sampler is repeated.

In Section IV.2, two heuristics were introduced. The first, which I named VCM$^+$, failed to satisfy the boundary conditions. The second, VCM*, added a normalization and proved to be robust over all tested scenes. In the original publication [Jendersie and Grosch 2018] I used a hash grid implementation, which had a lower quality than the one proposed in Section III.1. Instead, I will use the interpolated octree in the following. In Section III.3 it was shown that this octree has the smallest error of all comparable density estimates.

VCM$^+$: $N^+_{R,i}$ Eq. (IV.3) p. 104
VCM*: $N^*_{R,i}$ Eq. (IV.4) p. 104

Density Octree Sec. III.2 p. 89

Using the sampler variances directly yields a different heuristic which does not need arbitrary normalization factors. Like the previous heuristic it requires a density estimate. However, in the Ray Tracing Gem article [Jendersie 2019b] I replaced the density data structure with a direct footprint estimate for the density. This reduces the memory consumption and improves performance. In the following, both the original density estimate (OVCM, [J19]) and the improved density estimate (IVCM, see Table IV.1 Thesis) derived in the previous section will be evaluated. The additional exponent $+c$ in the Gem article only produces a minor improvement and is not used here.

Variance-based factor N_R Eq. (IV.10) p. 108

IV.4.1 Quality: Equal Time

The first question is, if rendering of a realistic scenario can benefit from the proposed changes. For scenes without pathological merge cases the performance decreases due to the added overhead. This is the case for far distant light sources, where connection samplers are weighted higher than merges. However, many indoor scenarios use lamps which produce exactly the bad cases for merges.

Figure IV.13 demonstrates that the Villa scene exhibits this behavior. In the equal time comparison, VCM$^+$ and VCM* produce less noise than VCM, although they achieve only $\approx 93\%$ of the sample count. While not directly visible, IVCM and OVCM perform even better, since their computational overhead is smaller. OVCM reached the same sample count as VCM in this experiment. This means that OVCM does not reduce the effectiveness in scenes without problems much, but increases the robustness for indoor scenes. Especially, if a footprint estimate is computed anyway, the improvement comes almost for free.

IV.4.2 Quality: Variance

All of the proposed methods are equally biased. They have the common bias of photon gathering in the merges, which can be removed over time via

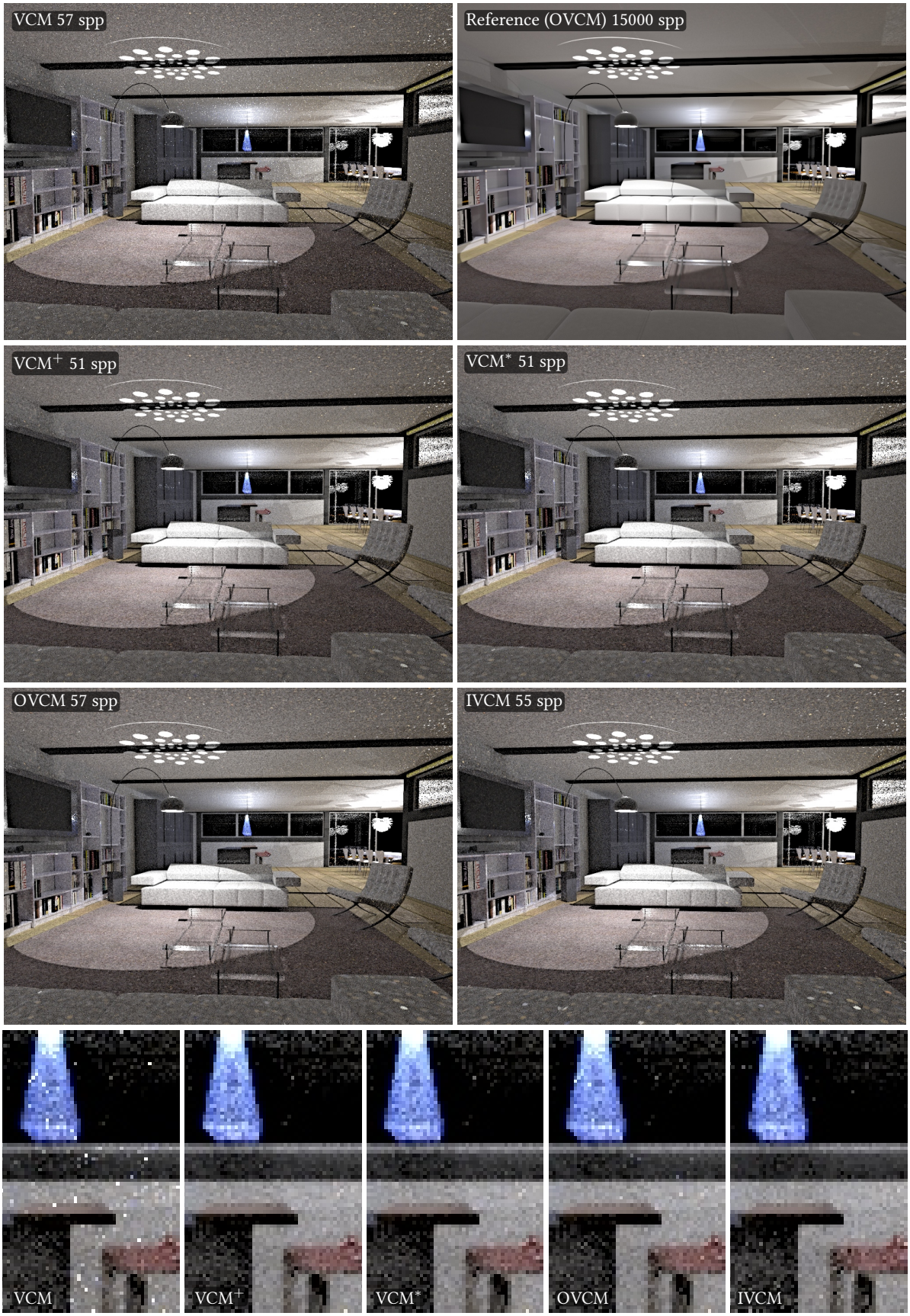


Figure IV.13: Equal time comparison (10 min) of the Villa scene (indoor view). VCM shows high variance noise on almost all surfaces whereas all of the improved variants successfully remove the noise caused by invalid merge weights.

progressive rendering. The changes to the reuse factor only influence how the different samplers are mixed. Any estimate of reuse factors is unbiased as long as all samplers use the same estimates which must be independent on the random decisions in the sampler. In this case the expected value of the sum of weights over all samplers is always one. Therefore, the only important error is the variance.

This variance can be measured by calculating the sample variance over the iterations. That means, each iteration provides a single sample per pixel, of which the variance can be accumulated in a second buffer. A method is better, if its converged variance image is darker. The optimum is a black image and only happens for ideal, variance-free samplers. Taking the average of the square root of the variance images gives the average standard deviation $\bar{\sigma}$ which will be given along with the images.

IV.4.3 Memory Consumption

The two first approaches VCM$^+$ and VCM* both required an additional data structure for the density queries. Using the octree from Section III.2 this boils down to 16 MiB additional memory – independent of scene or light path complexity.

In the footprint-based approaches OVCM and IVCM, additional memory is required per path vertex. To track the proposed footprints we need three numbers: $\sigma_{\boldsymbol{x}}$, $\sigma_{\triangleleft}$ and P. If storing all three numbers, only the footprints for the adjoint sub-path must be computed upon connection or merge. Alternatively, it is possible to recompute all footprints for the entire path which avoids storing $\sigma_{\boldsymbol{x}}$ and $\sigma_{\triangleleft}$. P must be stored in any case or inferred from some other information about the Russian roulette decisions.

$\sigma_{\boldsymbol{x}}, \sigma_{\triangleleft}$ Tab. IV.1 p. 110
P product of discrete event probabilities Eq. (IV.11) p. 109

In both cases, the total memory consumption depends on the number of stored vertices. This in turn depends on the path complexity and sample count and is typically in 2–10 millions. Per one million vertices 11.4 MiB and 3.8 MiB are required respectively if using 32 bit floats. The implementation used here stores all three values.

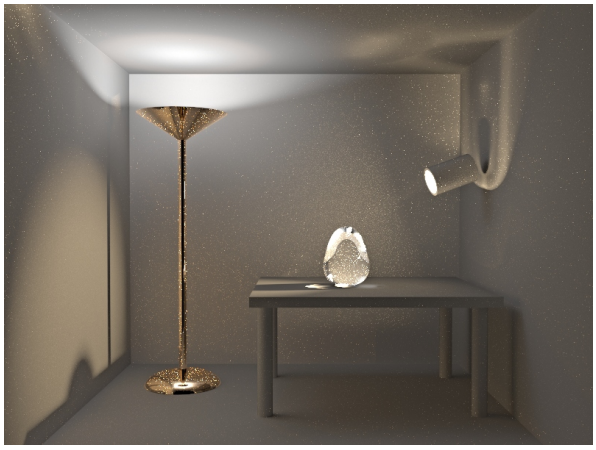
Summarizing, the footprint-based approaches may need more memory than the octree-based ones for large resolutions. However, some kind of footprint might be estimated and stored anyway for reasons of anti-aliasing [Akenine-Möller et al. 2019; Igehy 1999; Suykens and Willems 2001] or adaptive reconstruction [Belcour et al. 2013; Schjøth et al. 2007]. In this case, or if all values are computed on the fly, there is no additional memory overhead.

IV.4.4 Runtime

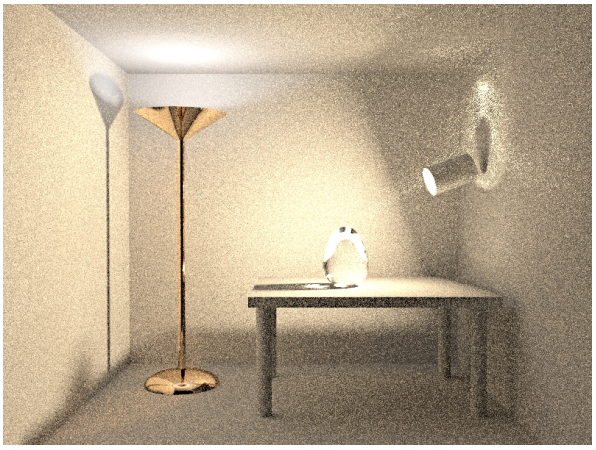
The main benefit for the footprint-based methods is their low computational overhead. As can be seen in Table IV.3, OVCM is only slightly slower than vanilla VCM (1.2% on average), but has considerably less variance.

The newer footprint estimate in this thesis does not pay off in this use case. It increases the cost while having no systematic improvement opposed to previous heuristic.

While VCM* combined with the new density octree data structure gives the best quality in most scenes, it is also the most expensive method. It



(a) BPT; $\bar{\sigma} = 0.318$



(b) VCM [GKDS]; $\bar{\sigma} = 0.649$

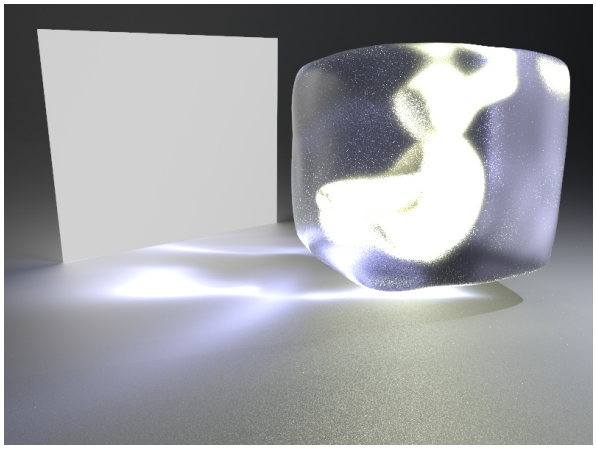
(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.237$

(b) VCM* [JG18]; $\bar{\sigma} = 0.214$

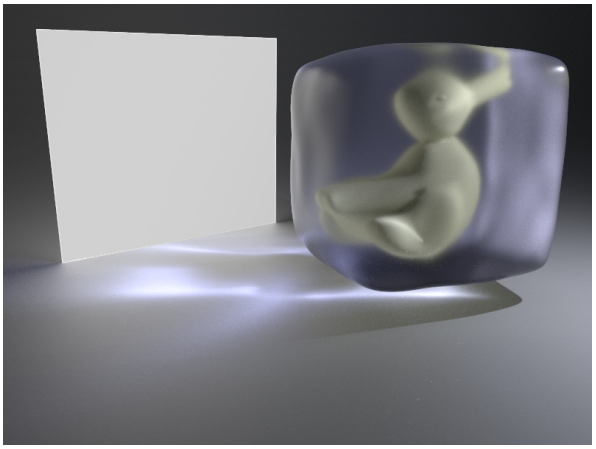
(a) OVCM [J19]; $\bar{\sigma} = 0.226$

(b) IVCM; $\bar{\sigma} = 0.215$

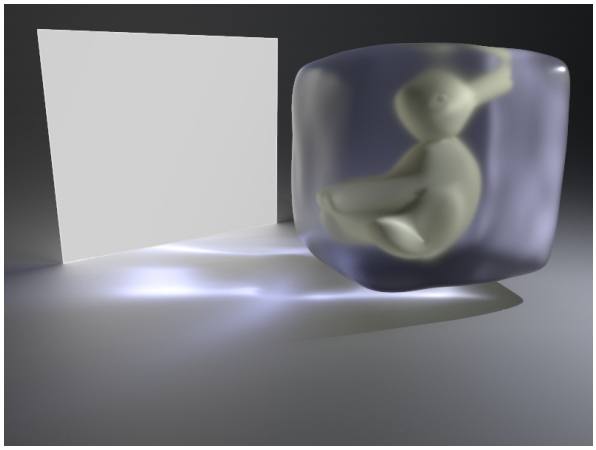
Figure IV.14: Standard deviation in the Veach-Bidir scene recorded over 10-70k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.



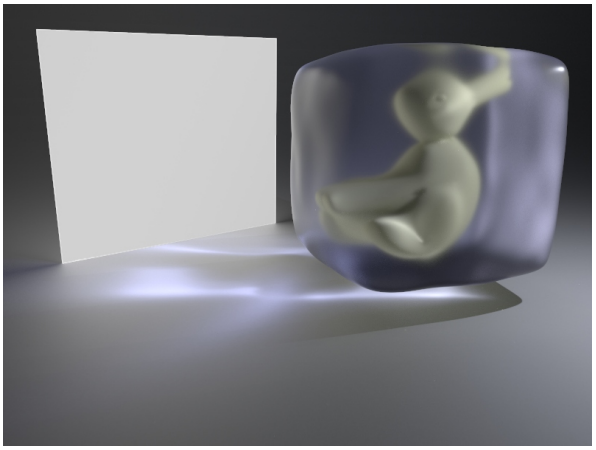
(a) BPT; $\bar{\sigma} = 1.234$



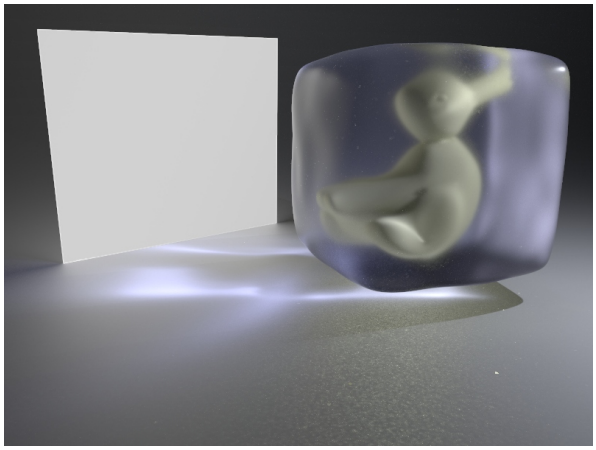
(b) VCM [GKDS]; $\bar{\sigma} = 0.478$



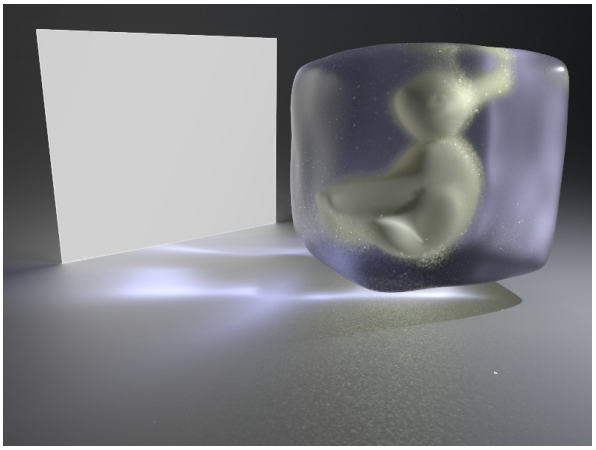
(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.451$



(b) VCM* [JG18]; $\bar{\sigma} = 0.442$



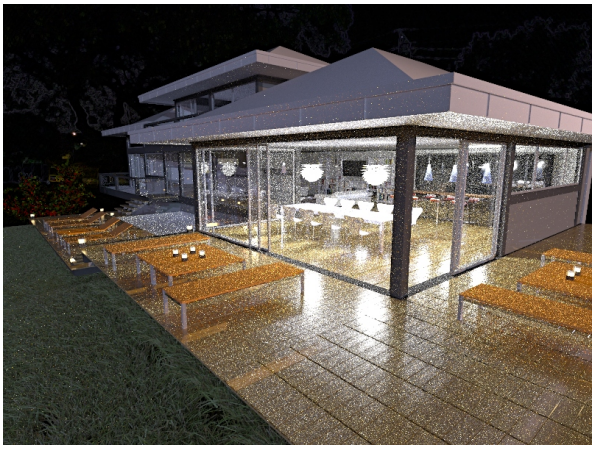
(a) OVCM [J19]; $\bar{\sigma} = 0.463$



(b) IVCM; $\bar{\sigma} = 0.469$

Figure IV.15: Standard deviation in the BunnyDuck scene recorded over 10-245k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.

(a) BPT; $\bar{\sigma} = 1.517$



(b) VCM [GKDS]; $\bar{\sigma} = 1.464$

(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.668$

(b) VCM* [JG18]; $\bar{\sigma} = 0.736$

(a) OVCM [J19]; $\bar{\sigma} = 0.696$

(b) IVCM; $\bar{\sigma} = 0.791$

Figure IV.16: Standard deviation in the Villa scene recorded over 30k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.

(a) BPT; $\bar{\sigma} = 0.363$

(b) VCM [GKDS]; $\bar{\sigma} = 0.278$

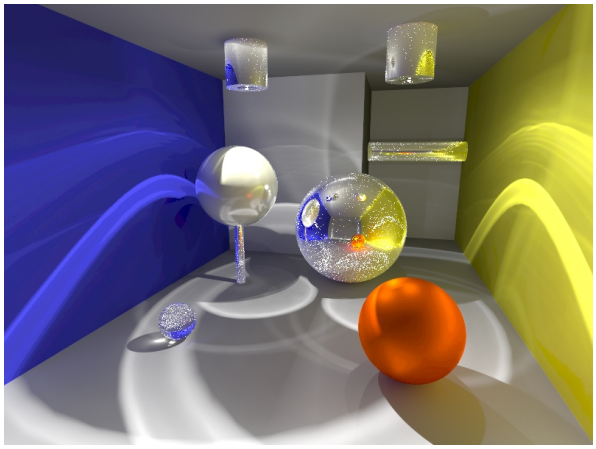
(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.259$

(b) VCM* [JG18]; $\bar{\sigma} = 0.258$

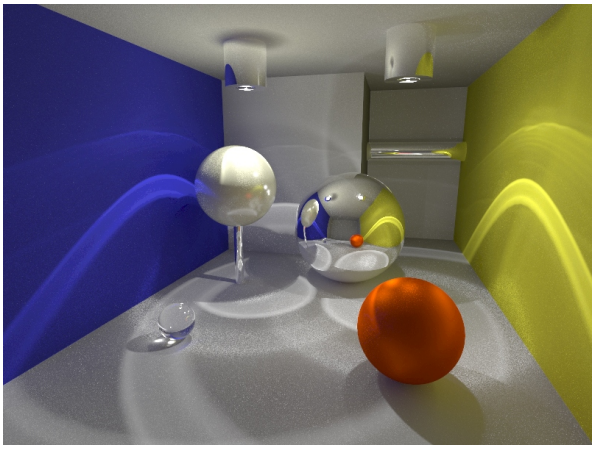
(a) OVCM [J19]; $\bar{\sigma} = 0.266$

(b) IVCM; $\bar{\sigma} = 0.269$

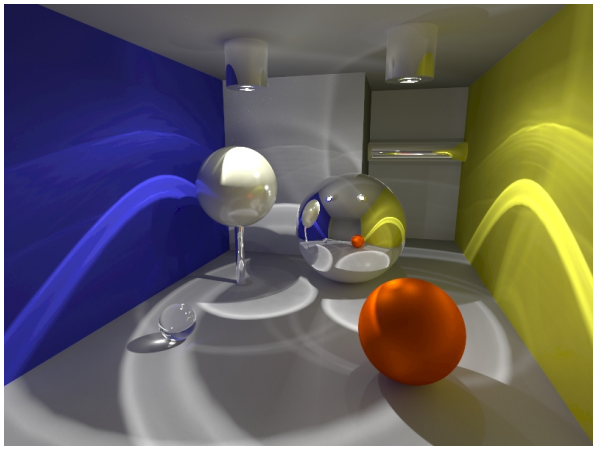
Figure IV.17: Standard deviation in the Bathroom scene recorded over 11k-30k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.



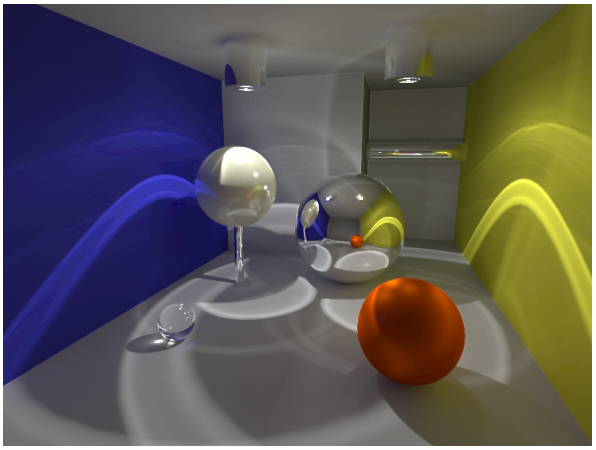
(a) BPT; $\bar{\sigma} = 1.014$



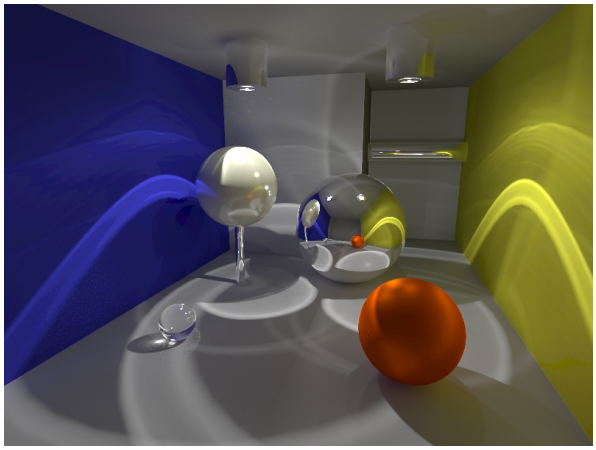
(b) VCM [GKDS]; $\bar{\sigma} = 0.306$



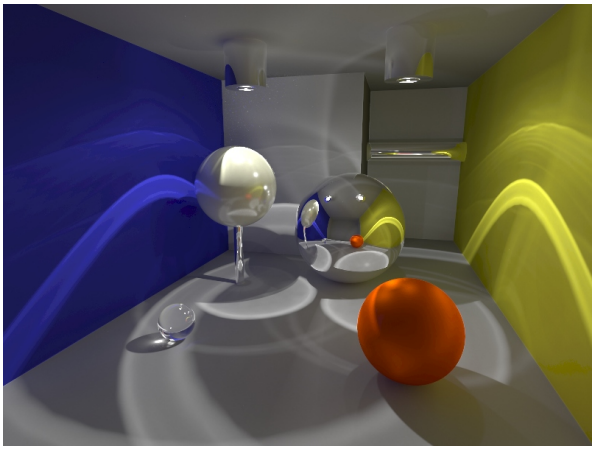
(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.285$



(b) VCM* [JG18]; $\bar{\sigma} = 0.275$



(a) OVCM [J19]; $\bar{\sigma} = 0.275$



(b) IVCM; $\bar{\sigma} = 0.275$

Figure IV.18: Standard deviation in the Mirrorballs scene recorded over 10k-40k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.

(a) BPT; $\bar{\sigma} = 0.619$

(b) VCM [GKDS]; $\bar{\sigma} = 0.522$

(a) VCM$^+$ [JG18]; $\bar{\sigma} = 0.518$

(b) VCM* [JG18]; $\bar{\sigma} = 0.518$

(a) OVCM [J19]; $\bar{\sigma} = 0.529$

(b) IVCM; $\bar{\sigma} = 0.534$

Figure IV.19: Standard deviation in the Sponza scene recorded over 30k iterations. Darker is better. $\bar{\sigma}$ gives the average over all pixels where again smaller is better.

Table IV.3: Processing time for 16 iterations with different VCM heuristics. The method VCM^*_{HG} uses the hash grid instead of the octree.

	VCM	VCM^+	VCM^*	VCM^*_{HG}	OVCM	IVCM
Veach-Bidir	303 s	395 s	396 s	335 s	310 s	319 s
BunnyDuck	138 s	205 s	209 s	158 s	139 s	144 s
Villa	1249 s	1477 s	1462 s	1327 s	1259 s	1265 s
Bathroom	1150 s	1265 s	1265 s	1204 s	1160 s	1183 s
MirrorBalls	1481 s	1802 s	1806 s	1579 s	1507 s	1609 s
Sponza	904 s	1024 s	1025 s	1009 s	909 s	934 s

is 24.1% slower than VCM on average. Additionally, it has to be said that the test renderer had a bad acceleration structure which means that the gap becomes even larger in a well optimized renderer.

Since the previously published implementation [Jendersie and Grosch 2018] used the cheaper hash grid, the timings for VCM^*_{HG} are given for reference. With 9% on average, the computational overhead is still large while the grid may scale badly with scene size. For the Sponza scene it is necessary to allocate a 200 MB hash map in order to avoid an overflow, because the reachable scene surface is large compared to the merge area.

Density Hash Grid Sec. III.1 p. 84

IV.4.5 Summary

Simply using the number of photons N_Φ in the MIS heuristic causes variance problems due to an underestimation of the event's variance. All four proposed methods reduce the noise reliable, where VCM^* has the highest quality in many cases. The footprint based OVCM is the cheapest of all methods and has still very good values. Therefore, I recommend this method for practical applications.

Trying to enhance the footprint estimates in this thesis did not yield the desired improvements. While I think that good and fast footprint estimates have many applications (the proposed heuristic, adaptive reconstruction, anti-aliasing, ...), I would not rely on the experimental formulas here. Especially the curvature terms are troublesome since their macroscopic nature breaks with assumptions of local density changes.

Chapter V
Microfacet Regularization for Glossy Paths

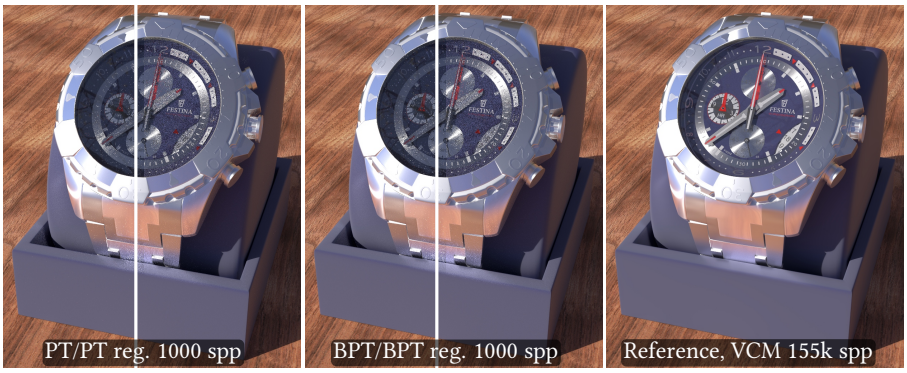


Figure V.1: A wrist watch with many glossy interactions (model courtesy of heraSK). Both regularized variants cover more of the light effects, although still noisy.

Even with the improved heuristics from the previous chapter, VCM will produce high variance noise for specific situations, some of which are:

1. Low visibility: The most extreme example is a room completely illuminated by light falling through a key hole.
2. Glossy paths: Too smooth, but not yet specular, surfaces can cause high variance dependent on the available samplers.
3. Far distant caustics: The photon density is much smaller than desired due to a high distance of the light source.

Problem one is not covered in this thesis, whereas the other two are the topic of this and the next chapter. So-called guidance methods reduce the variance in all three cases by learning the importance and radiance distribution in the scene over time [Herholz et al. 2016; Müller et al. 2017]. Later samples are then guided into the direction of the learned adjoint function. Hence, the PDF comes closer to the integrand which reduces the variance. Herholz et al. [2016] stored the functions as Gaussian mixture models. During sampling the BSDF is approximated by a Gaussian too. Now, a closed form product of the two functions can be computed, enabling a joint importance sampling. Müller et al. [2017] used a quadtree to store directional distributions, which is simpler and more robust than Gaussian mixtures. On the other hand, either the BSDF or the quadtree is sampled in an MIS

Variance of Monte Carlo II.2.2 p. 20

framework, which is less effective than the *product importance sampling* from Herholz et al.

Although the guidance methods reduce variance, they have problems with high frequency situations, because they need to smooth the signal in expectancy of noise. Furthermore, they require training iterations before any guidance is possible at all. An alternative solution for problem two is discussed in this chapter and in the related publication [Jendersie and Grosch 2019]:

Microfacet Model Regularization for Robust Light Transport
Johannes Jendersie and Thorsten Grosch
In: Computer Graphics Forum (Proc. of EGSR) 38.4, pp. 39–47

The basic idea of path space regularization is to blur the BSDFs to decrease the variance on glossy or specular paths. The blurring of BSDFs is not only applicable to VCM but can also be applied in simpler transport methods like PT, as demonstrated in Figure V.1. A motivation to enable difficult light effects in simpler transport methods is that production renderers typically use PT variants because of their lower overhead [Bala 2018; P. Christensen et al. 2018; Fascione et al. 2017a; b; Georgiev et al. 2018].

The regularization concept was introduced by Kaplanyan and Dachsbacher [2013b]. They focused on the otherwise infeasible specular paths while we extend and evaluate the idea in the broader context of microfacet models. Our approach is to select a roughness value α such that the maximum value of the microfacet BSDF is below a given threshold.

Microfacet BRDF II.5.5 p. 49
Microfacet BTDF II.5.6 p. 53

Independent of the approach there are several issues with path space regularization which will be discussed in this chapter. For instance, the regularization obviously introduces bias. This means that regularization should be adapted locally, to blur BSDFs only where necessary. In the publication [Jendersie and Grosch 2019] we introduced two heuristics to reduce the introduced bias which will be explained in Section V.5.

A second issue is that, if we apply regularization while keeping the MIS weight unmodified, the result will be biased and noisy. The reason is that we modify the measurement contribution function. This change must be incorporated into the MIS weight as will be detailed in Section V.3. A similar MIS was used by Bouchard et al. [2013], who applied Kaplanyan and Dachsbacher's method in an MIS combined approach.

Contribution function
Eq. (II.72) p. 64

The properties for regularization techniques we would like to have are:

- Low Noise: Most importantly, variance must be reduced.
- Consistency: Bias by itself is an error we would like to minimize. If possible any remaining bias should vanish over time.
- Energy Preservation: Helps to keep the general brightness of the scene.
- Small Overhead: Ease of implementation and small performance impact.

V.1 Variance with and without Regularization

The first question I want to discuss in detail is the source of variance in our transport methods. On a macroscopic level, the variance stems from the discrepancy between path measurement contribution $f(\mathcal{P})$ and the used PDF $p(\mathcal{P})$. However, if we select a specific sampler it becomes interesting to see which of the terms causes how much variance. Like in Section IV.3.1 p. 108 we can factorize the variance of a sampler into several events.

Path measurement contribution f Eq. (II.72) p. 64

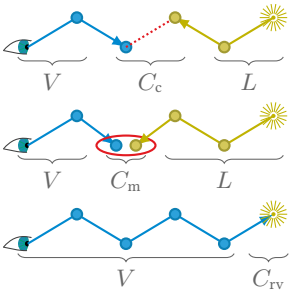
Each step of a random walk is one event in the form $\wp_i/p_i$. Additionally, there are different events for the connection, merge and random hit events. Like in the previous chapter we summarize the non-sampled terms with C, which are

Random hit II.7.3 p. 65
Connections II.7.4 p. 67
Merges II.7.5 p. 68

$$\text{Connection:} \quad C_\text{c} = \frac{\wp_k \cdot \left|\cos^{\rightarrow}_k\right| \cdot V(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) \cdot \left|\cos^{\leftarrow}_{k+1}\right| \cdot \wp_{k+1}}{\left\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\right\|^2}$$

$$\text{Merge:} \quad C_\text{m} = \wp_k \cdot K(\boldsymbol{x}_k, \boldsymbol{x}_{k'})$$

$$\text{Random Hit:} \quad C_\text{rv} = L_e(\boldsymbol{x}_\ell, \boldsymbol{d}^{\leftarrow}_\ell),$$



$C_\text{c}, C_\text{m}, C_\text{rv}$ Eq. (IV.7) p. 107

repeated here for completeness.

For any selected event X of the sampler, we can measure its variance $\mathrm{V}[X|Y]$ and expected value $\mathrm{E}[X|Y]$ under the condition of all other events Y. In practice we could determine the sample variance of the isolated terms associated with X, while fixing path length and sampling algorithm.

Now, we assume that the relative variance of any event can be bounded:

$$\frac{\mathrm{V}[X|Y]}{\mathrm{E}[X|Y]^2} \leq \tau_X \tag{V.1}$$

which is what we will try with regularization in the next step. If the relative variance of each event is bound by τ_X, it can be shown that a path with ℓ segments has the bound

Proof in A.3.1 p. 197

$$\frac{\mathrm{V}[f(\mathcal{P})/p(\mathcal{P})]}{\mathrm{E}[f(\mathcal{P})/p(\mathcal{P})]^2} \leq \tau_\text{path} = \sum_{i=1}^{\ell+1} \binom{\ell+1}{i} \tau_X^i. \tag{V.2}$$

That means, if all vertices are bounded, the variance of the path depends exponentially on its length. This can be seen by applying the binomial formula

$$(1+\tau_X)^\ell = \sum_{i=0}^{\ell+1} \binom{\ell+1}{i} \tau_X^i = 1 + \tau_\text{path}.$$

We can also invert the problem of determining a per event bound by fixing the bound τ_path. Distributing the threshold equally among all vertices, it is possible to compute a bound τ_X for each vertex such that the path's relative variance is below a target value:

$$\tau_X = (1+\tau_\text{path})^{\frac{1}{\ell}} - 1 \tag{V.3}$$

This gives us a useful parameter and insight on how to modify the individual events for variance reduction.

Restricting the relative variance is preferable since our perception is relative too (Weber-Fechner law [Fechner 1858; Weber 1834]). In dark areas we perceive the same absolute level of noise stronger than in bright areas. Also, it has to be expected that the results of a rendering process are tone-mapped, i.e. that the brightness is remapped after the end of the simulation to match our perception. In those cases, a relative measure restricts the error in a more meaningful way. Last but not least, the relative bound yields a much cleaner formulation for the bound of the path.

In the previous chapter we already observed that sampling events on a path probably have a low variance which we expressed with

Sampler variances IV.3.1 p. 108

$$\mathrm{V}\left[\prod_i \frac{\wp_i}{p_i}\right] = \varepsilon \approx 0.$$

Grouping the events on a path produced

$\mathcal{P}$ Sec. II.7.2 p. 63
V, C, L Sec. IV.3 p. 107

$$\begin{aligned}\mathrm{V}[\mathcal{P}] &= \mathrm{V}[V]\,\mathrm{E}[CL]^2 + \mathrm{E}[V]^2\,\mathrm{V}[CL] + \mathrm{V}[V]\,\mathrm{V}[CL] \\ &= \mathrm{V}[V]\,\mathrm{E}[C]^2\,\mathrm{E}[L]^2 + \mathrm{E}[V]^2\,\mathrm{V}[C]\,\mathrm{E}[L]^2 + \mathrm{E}[V]^2\,\mathrm{E}[C]^2\,\mathrm{V}[L] \\ &\quad + \mathrm{V}[V]\,\mathrm{V}[C]\,\mathrm{E}[L]^2 + \mathrm{V}[V]\,\mathrm{E}[C]^2\,\mathrm{V}[L] + \mathrm{E}[V]^2\,\mathrm{V}[C]\,\mathrm{V}[L] \\ &\quad + \mathrm{V}[V]\,\mathrm{V}[C]\,\mathrm{V}[L]\end{aligned}$$

which reduces to

$$\mathrm{V}[\mathcal{P}] \approx \mathrm{E}[V]^2\,\mathrm{V}[C]\,\mathrm{E}[L]^2 \tag{V.4}$$

under the assumption of well behaved samplers ($\mathrm{V}[V] = \mathrm{V}[L] = 0$).

In case there are weak importance sampling methods, the variance in those sampling events can be limited by clamping the throughput using $\min(c_{\max}, \wp_i/p_i)$ for some user selected $c_{\max}$. A good choice for $c_{\max}$ is the integrated value $|\Omega|^{-1}\int_\Omega \wp(\boldsymbol{d})\cos\theta\,\mathrm{d}\omega$ of the event multiplied with a factor slightly larger than one. For materials this is the directional albedo ϱ which perfectly makes sense. An ideal sampler with $p \propto \wp$ will always return this value. Using an albedo-relative bound is a good idea for reasons of perception. In any case, limiting the throughput will reduce the variance at the cost of energy loss.

ϱ Eq. (II.39) p. 40

However, if the assumption holds, the main source of variance is the term C from the non-sampled events of the path. This is the point where regularization comes into play. Obviously, clamping these values, too, will reduce the absolute variance *and* the expected value of the path. This means that energy simply vanishes and that the high variance events are masked out from the simulation. Contrarily, blurring the functions $\wp$ will preserve the energy while also reducing the variance.

In general, the variance of a random variable depends on its maximum value M and minimum value m given some particular PDF, as shown in the two upper bounds [Bhatia and Davis 2000; Popoviciu 1935]

$$\text{Popoviciu:}\quad \sigma^2 \le \frac{1}{4}(M-m)^2 \tag{V.5}$$

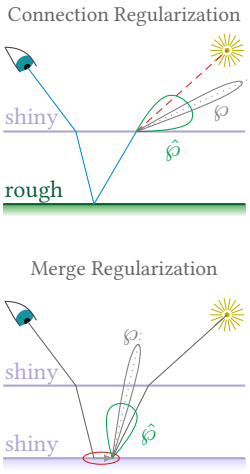
$$\text{Bhatia-Davis:}\quad \sigma^2 \le (M-\mu)(\mu-m) \tag{V.6}$$

where μ is the expected value of the function. Thereby, the Bhatia-Davis bound is stronger than Popoviciu's inequality.

As long as a regularization technique keeps the integrated value of a function constant, we can transfer the bounds to our problem. By blurring a BSDF we most likely reduce M and increase m and thus can expect a quadratic improvement of the variance. Note that the PDF under which an event X is sampled, depends on all other events Y. Since we do not know this PDF we cannot guarantee a reduction of M or an increase of m.

Unfortunately, the only way to reduce the variance of C_{rv} is to clamp the value of L_e which we will not do in the following. Also, the term C_{c} contains further functions like visibility V, the cosines and the distance $\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|$ which we cannot or will not modify in the following. Note that adding a constant or clamping the distance term is a regularization which is often applied to *Virtual Point Light* (VPL) rendering.

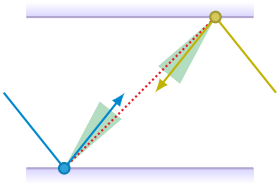
To summarize this section, we have learned that often variance manifests in the evaluated terms C of a sampler and not in the sampled parts. For those evaluated terms we are going to blur the functions $\wp$ (obtaining $\hat{\wp}$) which most likely reduces the variance. This also means that we can keep sampling the original functions, because an increased diffusion in the random walk would not reduce variance at all. We only have to blur functions in the connection or merge terms and never during random walk! However, since variance may also be caused by the other terms, this reduction cannot guarantee bounds for the final result.



V.2 REGULARIZATION TECHNIQUES

Our general goal is to reduce the variation of a function by limiting its maximum value under the condition of energy preservation. There are still multiple possibilities to smooth out the function $\wp$ which are introduced in this section. The solutions for specular- and roughness-based regularization only apply to selected functions $\wp$ whereas the virtual merge strategy applies to all possible functions.

V.2.1 SPECULAR REGULARIZATION



The first technique is that of Kaplanyan and Dachsbacher [2013b] for specular events. In specular reflections or refractions the exitant direction is deterministic and the corresponding BSDF is a Dirac impulse. Any connection or merge with such a function has infinite variance and will never be able to sample this event. Now, the idea of Kaplanyan and Dachsbacher is to accept any connection within a small cone around the perfect exitant direction.

Upon acceptance a contribution value is computed. For the value of the BSDF a uniform distribution over the cone is assumed, which yields

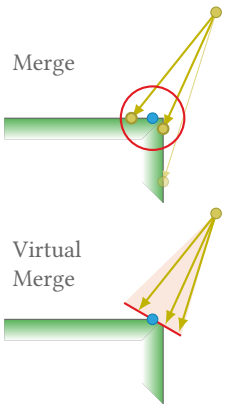
$$\rho_{\text{cone}} = \frac{1}{\omega_{\text{cone}}} = \frac{1}{2\pi(1-\cos\theta)} \tag{V.7}$$

To parametrize the cone opening angle, Kaplanyan and Dachsbacher decided to borrow the radius parameter from photon merges: $\theta = \arctan(r_i/d)$. To achieve consistency, this radius is shrunk over the iterations i. Merges themselves can be seen as a regularization strategy and show the same asymptotic behavior. Hence, a regularization with a single vertex can use the same optimized[1] sequence $r_i = r_0 \cdot i^{-1/6}$. The optimal[1] radius sequence for a regularization with two vertices is $r_i = r_0 \cdot i^{-1/12}$.

Consistent merge radius r_i
Eq. (II.90) p. 78
Length of connection segment d

V.2.2 THE VIRTUAL MERGE APPROACH

To generalize the technique for arbitrary functions, Kaplanyan and Dachsbacher already proposed the *virtual merge* approach. The idea is to importance sample the BSDF and then accept a connection if the sampled direction lies within the cone. To decide whether a vertex needs to be regularized or not, we can compare the value $\max(\rho)$ to ρ_{cone}. Only if $\max(\rho)$ is larger, regularization in this form makes sense at all. In detail the algorithm is:



1. Sample the BSDF $\rightarrow$ sample s with $\rho_s, p_s, \boldsymbol{d}_s$
2. Compare angle between $\boldsymbol{d}_s$ and the connection direction $\boldsymbol{d}_c$
   (a) If within cone return $\rho_s/(p_s \cdot \omega_{\text{cone}})$
   (b) Otherwise discard connection

This approach is statistically correct and is almost the same thing as a merge event, hence the name *virtual merge*. In both cases ρ_s/p_s describes the sampling event at the respective vertex. Solely the search space for the two elements differs. In a merge we can only find particles in the local area

[1] Optimized with respect to *Asymptotic RMSE* which contains bias and variance

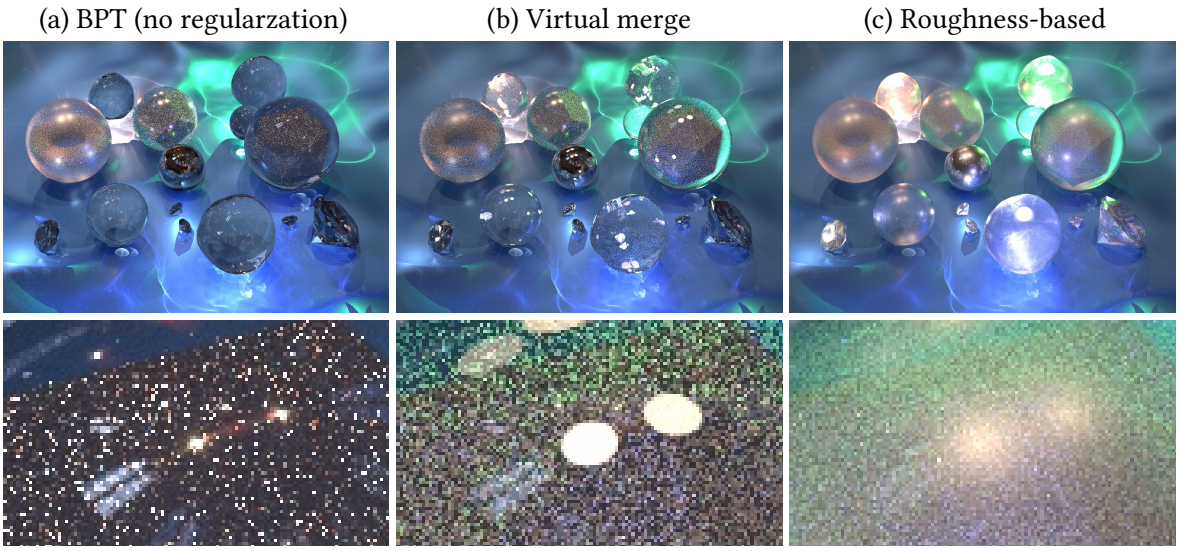


Figure V.2: Roughness-based reg. compared to virtual merges (BPT at 1000 spp). The image shows a folded cloth with several gems, metal and glass spheres, the latter partially containing diffuse objects. The three wobbly glass spheres contain point light sources. Further illumination comes from an environment map and three additional point lights.

πr^2 while the above strategy finds all events inside a cone. This is slightly more robust in the vicinity of edges.

In step 2.(a) of the algorithm, $p_s \cdot \omega_{\text{cone}}$ is a sample of the acceptance probability p_{acc}. If we were able to compute the correct probability $p_{\text{acc}} = \int_{\omega_{\text{cone}}} p(\boldsymbol{d})\,\mathrm{d}\omega$ and the integrated BSDF ρ_{int} over ω_{cone}, we could directly get the contribution value $\rho_{\text{int}}(\boldsymbol{d}_c)/p_{\text{acc}}(\boldsymbol{d}_c)$ without the random process. This technique is equivalent to a convolution of the BSDF with a box filter. Unfortunately, this is not possible in most situations in practice.

As can be expected, the additional random process introduces noise. The image in Figure V.2 (b) shows an example (with correct MIS). Especially in tail regions of glossy vertices, the noise becomes very large.

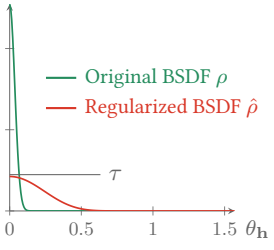
V.2.3 The Roughness-based Approach

To avoid the additional noise from the random event in *virtual merges* we can blur the BSDF by modifying its parameters. The respective parameter α is often called roughness. We want to select an $\hat{\alpha}$ such that

$$\forall \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow : \rho(\hat{\alpha}, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) \leq \tau \tag{V.8}$$

where τ is a threshold parameter. The regularized BSDF is then

$$\hat{\rho} = \rho(\max(\alpha, \hat{\alpha}), \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow), \tag{V.9}$$



where using $\max(\alpha, \hat{\alpha})$ guarantees that we only apply regularization if necessary. If we have an anisotropic model, both roughness parameters can be set individually with $\max(\alpha_x, \hat{\alpha})$ and $\max(\alpha_y, \hat{\alpha})$. This can change the degree of anisotropy, but is still a good choice. Details are given in Section V.2.3.

Figure V.2 (c) shows that the roughness-based approach is less noisy but also appears much more blurry than the virtual merge technique. This issue is reduced by hiding the bias heuristically, as will be done in Section V.5.

To solve Equation (V.8) we need to find the maximum $\bar{\rho}$ over all directions and invert it for alpha

$$\hat{\alpha} = \bar{\rho}^{-1}(\tau).$$

The function $\bar{\rho}$ depends on the model and might not be available (unbounded) or invertible. In these cases we have to fall back to an approximative solution.

Invertible Bound for Microfacet Models

Microfacet models are the most commonly used primitives for material descriptions. They were introduced in Sections II.5.5 and II.5.6 and consist of three functions D, G, F and a few other terms. Maximizing all terms together is infeasible because there are four degrees of freedom $\Theta = (\theta_\downarrow, \phi_\downarrow, \theta_\uparrow, \phi_\uparrow)$ (two angles per direction $\boldsymbol{d}$). We would need to solve $\partial\rho/\partial\Theta = 0$ which is a system with four trigonometric equations. Therefore, we will maximize individual terms in the following.

The Fresnel term F or $(1 - F)$ respectively is bounded by one and independent of α. Setting it to one maximizes the BSDF in a conservative way and worked well in practice.

The microfacet distribution D is usually maximized for $\boldsymbol{h} = \boldsymbol{n}$ as can be shown via derivation. An exception is the Beckmann-Spizzichino distribution D_{BS}. For roughness values $\alpha > 1/\sqrt{2}$ it has a maximum at $\langle \boldsymbol{h}, \boldsymbol{n} \rangle = 1/\alpha\sqrt{2}$. Also see Appendix A.3.4 for details. However, for practical applications with bounded α values (typically $\alpha \in [0, 1]$) this anomaly does not matter. In this case the maximum value of all microfacet distribution functions is

D_{BS} Eq. (II.56a) p. 51

$$\bar{D}(\alpha) = \frac{1}{\pi\alpha^2}. \tag{V.10}$$

The next term to be maximized is the geometric term

$$\bar{G}(\alpha) = \max\left[\frac{G(\alpha, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)}{|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n} \rangle \cdot \langle \boldsymbol{d}_\uparrow, \boldsymbol{n} \rangle|}\right]$$

where we include the shared denominator between reflection and transmission model, because G and the denominator cancel each other in certain configurations.

Microfacet BRDF II.5.5 p. 49
Microfacet BTDF II.5.6 p. 53

When using the V-cavity shadowing model [Torrance and Sparrow 1967] the shadowing is independent of the roughness parameter. It is maximized if the two directions are opposite each other with respect to the normal. In this case the above terms are unbounded and become infinity at grazing angles due to the denominator $|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n} \rangle \cdot \langle \boldsymbol{d}_\uparrow, \boldsymbol{n} \rangle|$. Our best option is to set

V-cavity Eq. (II.54) p. 50

$$\bar{G}_{\text{V}} = 1. \tag{V.11}$$

For the Smith model [Smith 1967] the shadowing $G(\alpha, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow) = G_1(\alpha, \boldsymbol{d}_\downarrow) \cdot G_1(\alpha, \boldsymbol{d}_\uparrow)$ depends on the roughness and compensates the denominator. Let $\theta = \arccos(\langle \boldsymbol{d}, \boldsymbol{n} \rangle)$ be the angle between the normal and the respective

direction. Then the bounds are

$$G_{1\mathrm{GGX}}(\alpha,\theta) = \frac{2}{1+\sqrt{1+\alpha^2\tan^2(\theta)}}$$
$$\lim_{\theta\to\pi/2}\frac{G_{1\mathrm{GGX}}(\alpha,\theta)}{\cos(\theta)} = \frac{2}{\alpha} \tag{V.12}$$

$$G_{1\mathrm{BS}}(\alpha,\theta) = \frac{2}{1+\mathrm{erf}\left(\frac{1}{\alpha\tan(\theta)}\right)+\frac{\alpha\tan(\theta)}{\sqrt{\pi}}\exp\left(\frac{1}{\alpha\tan(\theta)}^2\right)}$$
$$\lim_{\theta\to\pi/2}\frac{G_{1\mathrm{BS}}(\alpha,\theta)}{\cos(\theta)} = \frac{2\sqrt{\pi}}{\alpha} \tag{V.13}$$

for each of the two parts $G_1(\alpha,\boldsymbol{d}_\downarrow)/\langle\boldsymbol{d}_\downarrow,\boldsymbol{n}\rangle$ and $G_1(\alpha,\boldsymbol{d}_\uparrow)/\langle\boldsymbol{d}_\uparrow,\boldsymbol{n}\rangle$. The above shadowing terms are taken from the work of Walter et al. [2007]. The authors also suggested to use the Beckmann shadowing term for the cosine model $G_{1\mathrm{Cos}} = G_{1\mathrm{BS}}$ due to the high similarity of the two functions.

The remaining terms which need to be maximized are

$$\bar{R}(\alpha) = \max\left[\frac{1}{4}\right] = \frac{1}{4}$$

which appears in the reflective BRDF and Microfacet BRDF II.5.5 p. 49

$$\bar{T}(\alpha) = \max\left[\frac{\eta_t^2\,|\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle\cdot\langle\boldsymbol{d}_\uparrow,\boldsymbol{h}_t\rangle|}{(\eta_i\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle+\eta_t\langle\boldsymbol{d}_\uparrow,\boldsymbol{h}_t\rangle)^2}\right]$$

for the refractive BTDF. The numerator is bounded by $\max(\eta_i,\eta_t)^2$, because the two cosines will become at most one and the direction independent refraction index is simply the maximum of the two refraction indices. Microfacet BTDF II.5.6 p. 53

To maximize $\bar{T}$, the denominator must be minimized. Since the two direction vectors $\boldsymbol{d}_\downarrow$ and $\boldsymbol{d}_\uparrow$ are connected by Snell's law Snell's law Eq. (II.42) p. 42

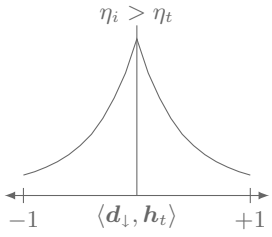
$$\langle\boldsymbol{d}_\uparrow,\boldsymbol{h}_t\rangle = -\mathrm{sign}(\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle)\sqrt{1-\frac{\eta_i^2}{\eta_t^2}\left(1-\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle^2\right)},$$

we can express the denominator with respect to a single cosine (here $\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle$)

$$\left(\eta_i\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle-\mathrm{sign}(\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle)\sqrt{\eta_t^2-\eta_i^2\left(1-\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle^2\right)}\right)^2$$

and minimize it. We find its minima at $\langle\boldsymbol{d}_\downarrow,\boldsymbol{h}_t\rangle = \{-1,1\}$ with values $(-\eta_i+\eta_t)^2$ and $(\eta_i-\eta_t)^2$. Due to the square, we can ignore the sign and use the same term in both situations.

Denominator function



Putting everything together we get our invertible bounds

$$\bar{\rho}_r(\alpha) = \frac{1}{4\pi\alpha^2}\bar{G}(\alpha) \tag{V.14}$$

$$\bar{\rho}_t(\alpha) = \frac{\max(\eta_i,\eta_t)^2}{\pi\alpha^2(\eta_i-\eta_t)^2}\bar{G}(\alpha) \tag{V.15}$$

$$\text{with}\quad \bar{G}(\alpha) = \begin{cases} 1 & \text{V-cavity} \\ 4/\alpha^2 & \text{Smith, GGX} \\ 4/\pi\alpha^2 & \text{Smith, BS and Cosine} \end{cases} \tag{V.16}$$

This bound is not strict, since for the V-cavity model the maximum function value is always infinity independent of the roughness. Although we maximized all terms individually, a local maximum turns out to be at normal incidence for most cases. Only if the microfacet distribution does not have its maximum at the surface normal direction this changes. The global maximum seems to be at grazing angles in all cases.

Exploiting these observations we can easily generalize to other material models. It makes sense to evaluate the BSDF at normal incidence and to invert the resulting value for the roughness parameter. With this perspective we can always use $\bar{G}(\alpha) = 1$ independent of the chosen shadowing model, which simplifies the application to more complex materials.

The roughness based technique is not always energy preserving, because of the energy loss at high roughness values. Only if the energy loss is compensated, the approach is fully energy preserving. This however will only happen for very strong regularizations in which case the bias through regularization is the more severe problem.

Energy loss Sec. II.5.8 p. 55 and Sec. II.5.5 p. 49

ANISOTROPIC BSDFS

For anisotropic materials there are several possible choices how the minimum alpha $\hat{\alpha}$ is applied to the two parameters α_x and α_y. The simples option is to take the maximum for both parameters individually:

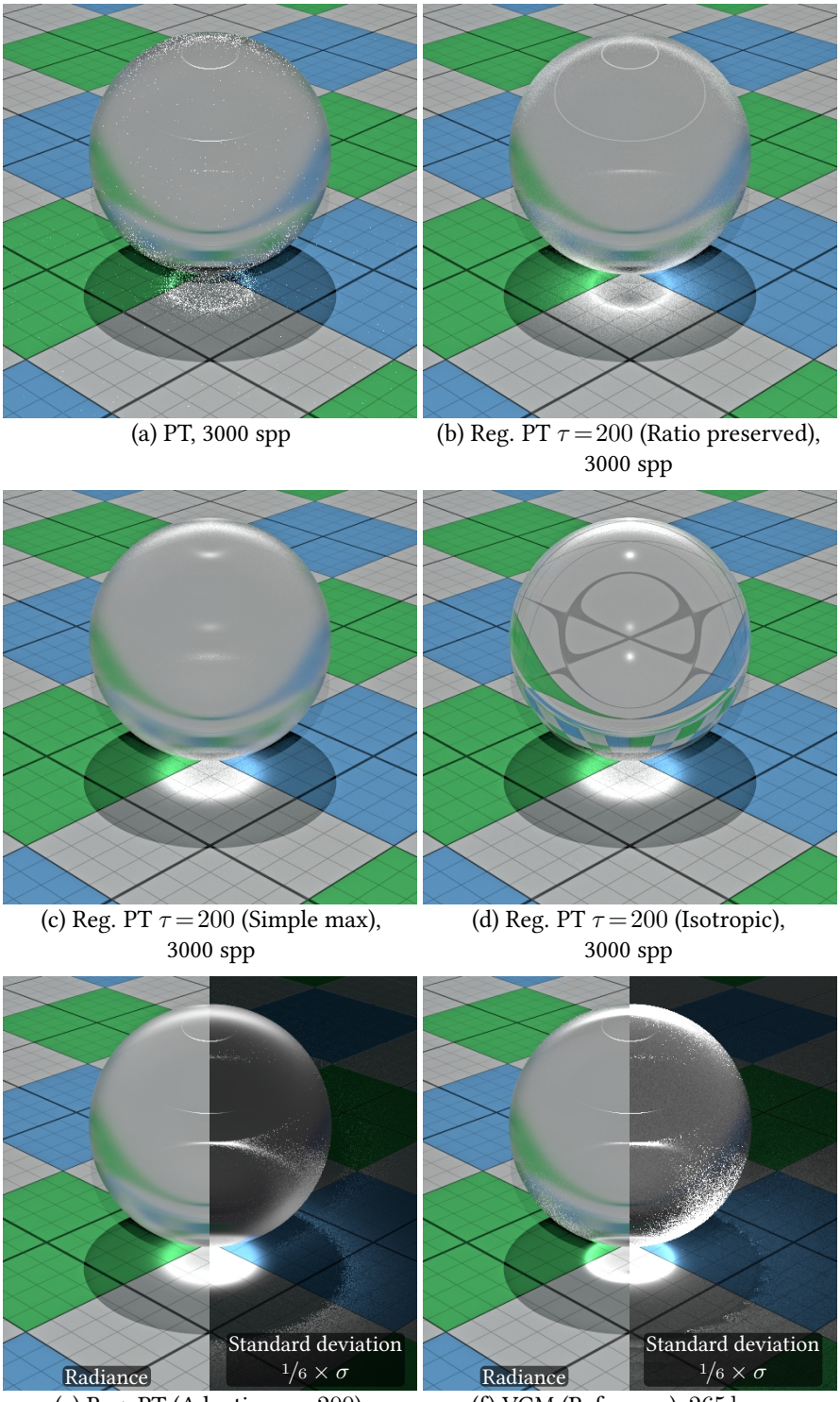
$$\alpha'_x = \max(\alpha_x, \hat{\alpha}), \tag{V.17a}$$
$$\alpha'_y = \max(\alpha_y, \hat{\alpha}). \tag{V.17b}$$

Another possibility would be to keep the ratio of anisotropy as long as possible:

$$\alpha'_x = \max\left(\alpha_x, \min(1, \hat{\alpha} \cdot R) \cdot \max(1, \hat{\alpha}/R)\right), \tag{V.18a}$$
$$\alpha'_y = \max\left(\alpha_y, \min(1, \hat{\alpha}/R) \cdot \max(1, \hat{\alpha} \cdot R)\right), \tag{V.18b}$$

where $R = \sqrt{\alpha_x/\alpha_y}$ is the ratio of anisotropy. Thereby the $\min(1, \square)$ guarantees that none of the parameters exceeds one. In that case the clamped part is applied to the other parameter instead ($\max(1, \square)$). The product of all four terms is always $\hat{\alpha}^2$ and thus should equal the total amount of regularization of the isotropic case.

There are two criteria which should be met by the anisotropy handling. On the one hand, the regularized variant should look similar to the reference. On the other hand, the noise level should be comparable to that of an isotropic model under the same threshold parameter τ. Figure V.3 demonstrates both strategies (images (b) and (c)). Interestingly, the simpler method from Equation V.17 is better with respect to both criteria. Using it in connection with the adaptive threshold τ' (image (e)), which will be introduced in Section V.5, we get very close to the reference. For the example scene, the regularized PT even outperforms VCM with respect to variance.



(a) PT, 3000 spp

(b) Reg. PT $\tau = 200$ (Ratio preserved), 3000 spp

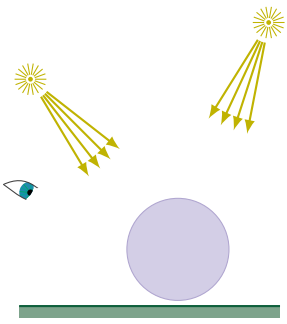
(c) Reg. PT $\tau = 200$ (Simple max), 3000 spp

(d) Reg. PT $\tau = 200$ (Isotropic), 3000 spp

(e) Reg. PT (Adaptive $\tau = 200$), 220 k spp

(f) VCM (Reference), 265 k spp

Figure V.3: Comparison of anisotropic regularization strategies (b) and (c) with non-regularized PT (a) and the isotropic case (d). The simple max strategy is more similar to the reference (f) than the ratio preserving strategy. Furthermore, its noise level is closer to that of the isotropic case (d) which is preferable, because different materials should react similar under the same parameter τ. Image (e) shows the converged state of an adaptive regularized PT (Sec. V.5), which has even less variance than VCM while still being very similar to the ground truth. The standard deviation is shown on the right of (e) and (f), where darker is better.



V.2.4 DISCUSSION: CONTROL VARIATE APPROACH

Virtual Merges V.2.2 p. 134

As stated in the merge approach section, we could reduce the noise, if we can compute p_{acc} correctly in a closed form. In fact there exists a closed form solution for the integral of the cosine function over a polygonal or spherical domain: Heitz et al. [2016a] used the transformation invariance of this integral to linearly transform polygonal light sources to approximate shapes of BSDFs. Dupuy et al. [2017] applied a similar technique to disc shaped integration regions. By using their approach we would get a good approximation of the searched probability. It is possible to use this approximation directly, but this would increase the bias.

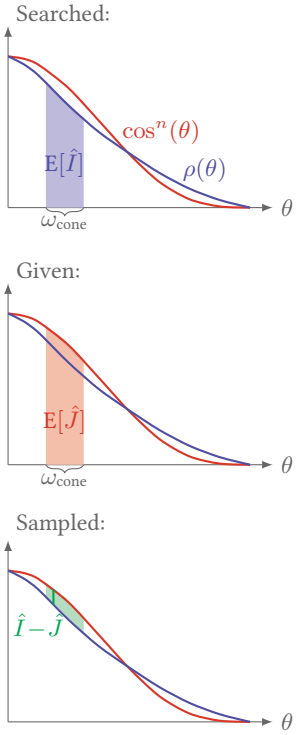
To compensate for the additional bias it is also possible to use the approximation as control variate [Hickernell et al. 2005; Kahn and Marshall 1953; Rothery 1982]

$$\hat{I}' = \hat{I} - c(\hat{J} - \mathrm{E}[\hat{J}]) \tag{V.19}$$

where $\hat{I} = p_s \cdot \omega_{\text{cone}}$ is the available estimator of the searched acceptance probability, $\hat{J}$ is the cosine value for the linear transformed direction $\boldsymbol{d}_s$, $\mathrm{E}[\hat{J}]$ is the integrated, but approximated, solution for p_{acc} and c is a constant factor which should be $c = -\mathrm{Cov}[\hat{I}, \hat{J}]\,/\mathrm{V}[\hat{J}]$ for an optimal variance reduction [Rothery 1982]. The estimator $\hat{J}$ is the control variate where we use the closed form solution [Dupuy et al. 2017] for its expected value.

The same control variate can be applied to obtain a lower noise estimate of ρ_{int} which is also required for a closed form virtual merge evaluation.

An advantage of the virtual merge and the control variate techniques is that peaks at grazing angles are regularized, too. With the cone-shaped convolution of BSDF and PDF in these two approaches, it is possible to include all peaks of arbitrary materials. In the roughness-based strategy this term was either unbounded or strongly dependent on the roughness.



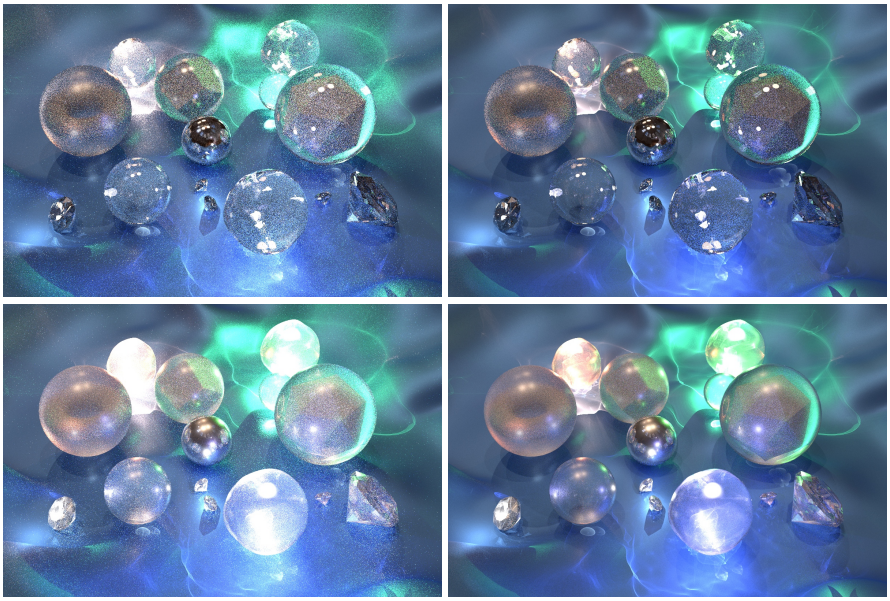


Figure V.4: Regularized BPT with the virtual merge technique (top row) and the roughness-based technique (bottom row). Using the path PDF MIS weights yields noise and too bright images (left column), which can be corrected (right column).

V.3 CORRECTING THE MIS WEIGHTS

Independent of the applied technique, regularization changes the estimator. If applying the usual MIS weights, based on path PDFs (Sec. II.8 p. 71) without modifications, the result is both more noisy and more biased. Figure V.4 demonstrates the effects of an invalid and a corrected MIS weighting. While more noise is expected, the bias due to the weights is somewhat surprising.

To understand this we can consider the example of pure specular transport paths. Any connection will form an invalid path with zero contribution. Using the relative weight computation scheme, the path PDFs of all other paths will be zero and the assigned weight becomes one. This is correct as long as the contribution is zero too. However, through regularization this contribution is now greater than zero. In other words: each sampler contributes a higher value than expected from looking onto path PDFs alone.

Relative MIS computation in BPT Eq. (II.88) p. 76

The reason for the poor behavior of the MIS weights is that the common assumption of a constant path measurement contribution function f is not valid in connection with regularization. Instead we have to use the estimator form of the MIS weight, which is derived in Section II.7.6. In each regularized connection event we change the BSDF of two vertices and for each regularized merge one BSDF is changed.

Path contribution f Eq. (II.72) p. 64

Estimator MIS weights Eq. (II.84) p. 69

To simplify the implementation in existing frameworks, it is also possible to include the changes made to f into the path probabilities. Therefore let

$$\hat{p}(\boldsymbol{d}_\uparrow) = \frac{\hat{\rho}}{\rho} \cdot p(\boldsymbol{d}_\uparrow) \tag{V.20}$$

be the corrected event PDF which includes the arbitrary changes of the regularized BSDF $\hat{\rho}$ over ρ. Since most terms in the ratio do not depend on

roughness α they cancel out and we get

$$\frac{\hat{\rho_r}}{\rho_r} = \frac{\hat{\rho_t}}{\rho_t} = \frac{D(\hat{\alpha}, \boldsymbol{h})G(\hat{\alpha}, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)}{D(\alpha, \boldsymbol{h})G(\alpha, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)} \tag{V.21}$$

for both microfacet models under consideration. If the V-cavity shadowing model is used, the G term also vanishes. In that case it is even possible to compute $\hat{\rho}$ directly, without computing the quotient in Equation (V.21) first, as shown in the following.

Using the visible microfacet sampling of Heitz and d'Eon [2014], the sampling probabilities for the two models are

$$\begin{aligned} p_r(\alpha, \boldsymbol{d}_\uparrow) &= \frac{D(\alpha, \boldsymbol{h})G(\alpha, \boldsymbol{d}_\downarrow)F(\boldsymbol{d}_\downarrow, \boldsymbol{h})}{4\left|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n}\rangle \langle \boldsymbol{d}_\uparrow, \boldsymbol{n}\rangle\right|} \\ &= D(\alpha, \boldsymbol{h})G(\alpha, \boldsymbol{d}_\downarrow)B_r \qquad \text{(V.22)} \\ \text{and } p_t(\alpha, \boldsymbol{d}_\uparrow) &= \frac{\eta_t^2 D(\alpha, \boldsymbol{h})G(\alpha, \boldsymbol{d}_\downarrow)(1 - F(\boldsymbol{d}_\downarrow, \boldsymbol{h}))}{(\eta_i \langle \boldsymbol{d}_\downarrow, \boldsymbol{h}_t\rangle + \eta_t \langle \boldsymbol{d}_\uparrow, \boldsymbol{h}_t\rangle)^2 \left|\langle \boldsymbol{d}_\downarrow, \boldsymbol{n}\rangle \langle \boldsymbol{d}_\uparrow, \boldsymbol{n}\rangle\right|} \\ &= D(\alpha, \boldsymbol{h})G(\alpha, \boldsymbol{d}_\downarrow)B_t \qquad \text{(V.23)} \end{aligned}$$

For brevity we can summarize all parametrization-independent terms (Fresnel and denominators) as B_r and B_t, respectively. Then we can insert any of the two PDFs and Equation (V.21) into Equation (V.20):

$$\begin{aligned} \frac{\hat{\rho}}{\rho} \cdot p(\alpha, \boldsymbol{d}_\uparrow) &= \frac{D(\hat{\alpha}, \boldsymbol{h}) \cdot G(\hat{\alpha}, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)}{\cancel{D(\alpha, \boldsymbol{h})} \cdot G(\alpha, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)} \cdot \cancel{D(\alpha, \boldsymbol{h})} \cdot G(\alpha, \boldsymbol{d}_\downarrow) \cdot B \\ &= \frac{G(\hat{\alpha}, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)}{G(\alpha, \boldsymbol{d}_\downarrow, \boldsymbol{d}_\uparrow)} \cdot D(\hat{\alpha}, \boldsymbol{h}) \cdot G(\alpha, \boldsymbol{d}_\downarrow) \cdot B \qquad \text{(V.24)} \end{aligned}$$

Due to independence of the V-cavity shadowing and the roughness, we can further cancel and replace terms such that

$$\begin{aligned} \text{(V.24), V-cavity} \Rightarrow D(\hat{\alpha}, \boldsymbol{h})G(\hat{\alpha}, \boldsymbol{d}_\downarrow)B \\ = p(\hat{\alpha}, \boldsymbol{d}_\uparrow). \end{aligned}$$

Hence, it is possible to evaluate p with respect to the regularized parameter $\hat{\alpha}$ instead of computing the fraction of BSDFs explicitly.

For the Smith model we cannot simplify the term in this way, but we can still reduce the overhead for a practical implementation

$$\begin{aligned} \text{(V.24), V-cavity} \Rightarrow \frac{G(\hat{\alpha}, \boldsymbol{d}_\downarrow) \cdot G(\hat{\alpha}, \boldsymbol{d}_\uparrow)}{\cancel{G(\alpha, \boldsymbol{d}_\downarrow)} \cdot G(\alpha, \boldsymbol{d}_\uparrow)} \cdot D(\hat{\alpha}, \boldsymbol{h}) \cdot \cancel{G(\alpha, \boldsymbol{d}_\downarrow)} \cdot B \\ = \frac{G(\hat{\alpha}, \boldsymbol{d}_\uparrow)}{G(\alpha, \boldsymbol{d}_\uparrow)} \cdot D(\hat{\alpha}, \boldsymbol{h}) \cdot G(\hat{\alpha}, \boldsymbol{d}_\downarrow) \cdot B \\ = \frac{G(\hat{\alpha}, \boldsymbol{d}_\uparrow)}{G(\alpha, \boldsymbol{d}_\uparrow)} \cdot p(\hat{\alpha}, \boldsymbol{d}_\uparrow). \end{aligned}$$

To conclude, we have to modify the calculated sampling probabilities of all events to get a correct MIS weight following the standard implementation. Note that we still sample the non-regularized original PDF during random walk. This is only an artificial change for the computation of the MIS weights.

V.3.1 MIS Weights for Virtual Merges

The virtual merge strategy samples a BSDF and accepts the connection if the sample is within the allowed cone of the connection direction. We already found that $p_s \cdot \omega_{\text{cone}}$ is a sample of the acceptance probability p_{acc}. It can be inserted into the path PDF for regularized connections used in the MIS computation.

Virtual merges Sec. V.2.2 p. 134

Fortunately, we can simplify the implementation of the virtual merge MIS. In the numerically robust form of the MIS weights, we always compute ratios of path densities. Hence, if $p(\boldsymbol{d}_\uparrow)$ was stored at a vertex, it is possible to store $p(\boldsymbol{d}_\uparrow)/p_{\text{acc}}$ instead, because the stored PDF is used for all but the current path. This leads to

$$\frac{p(\boldsymbol{d}_\uparrow)}{p_{\text{acc}}} = \frac{p(\boldsymbol{d}_\uparrow)}{p(\boldsymbol{d}_\uparrow) \cdot \omega_{\text{cone}}} = \frac{1}{\omega_{\text{cone}}} = \tau$$

using the single sample approximation of p_{acc} and $\tau = \rho_{\text{cone}}$. Also see the next section for the choice of τ.

ρ_{cone} Eq. (V.7) p. 134

This means that a connection with a random acceptance is similar to a continuation of the random walk. The difference is that the random walk is always accepted while the connection is only accepted with a probability of $1/\omega_{\text{cone}}$. Thus, we can replace the PDF p with $\hat{p} = \tau$ for regularized connections. For standard connections we have to keep p, which is similar to the changes made for roughness-based regularization. Note that this is not the optimal choice, because the true probability $p_{\text{acc}} = 1/\int_{\omega_{\text{cone}}} p(\boldsymbol{d})\,\mathrm{d}\omega$ is approximated by a single sample. The implicit assumption used here is that the sample is within the cone with a probability proportional to the size of the cone.

V.4 CONSISTENT PARAMETRIZATION

Until now, the choice of the threshold τ was left open. Smaller values of τ lead to more blurring and less variance. Preferring lower variance, we found values $\tau \in [10, 30]$ working equally well for all scenes with PT and BPT. However, if we use a better integration method like VCM, the threshold can be increased to around 1000.

In any case we would like to have a consistent estimator. We can use any series of an increasing τ to achieve that. At some finite point in time, the threshold will always be larger than the estimated bound $\bar{\rho}$ which disables the regularization for the event afterwards. The infinite number of later iterations are unbiased and are as consistent as the basic transport method (PT, ...) is.

Consistent Sec. II.2.1 p. 19

Also, we can relate τ to the known optimized rates of r_i to have a meaningful series. Therefore, we define $\tau = \rho_{\text{cone}}$ which unifies specular and roughness-based regularization. This gives us two possible options for the user determined initial parameter. The first (setting r_i)

r_i Eq. (II.90) p. 78

$$\tau_i = \frac{1}{2\pi(1 - \cos(\arctan(r_i/d)))} \tag{V.25}$$

depends on a segment length d. This causes problems for three reasons. First, the BSDF loses its reciprocity because the two adjacent segments to a vertex do not have the same length. Dependent on the transport direction, this would produce two different parametrizations. Second, the smoothness of the BSDF would change for different connection directions, leading to arbitrary shapes of highlights. And third, it does not apply to regularization in merge events.

To resolve these problems we can solve Equation (V.25) for r_0 and reinsert the term into the same equation:

$$\tau_i = \frac{1}{2\pi\left[1 - \cos\left(\arctan\left(\frac{\sqrt{4\pi\tau_0 - 1}}{2\pi\tau_0 - 1} \cdot i^{-\lambda}\right)\right)\right]} \tag{V.26}$$

where λ is either $1/6$ for merge regularization or $1/12$ for connection regularization. The derivation of Equation (V.26) is given in Appendix A.3.5.

Although the algorithm is consistent, it will not converge in a practical amount of time. In Figures V.5 and V.6 we can see that the results are very noisy and visibly biased even after a high number of iterations. However, the regularized results can be much closer to the ground truth as demonstrated by Figure V.6. We can also see that the consistent parametrization with a moderate initial τ_0 (e) is less biased than the non-consistent variant with a much higher threshold (d).

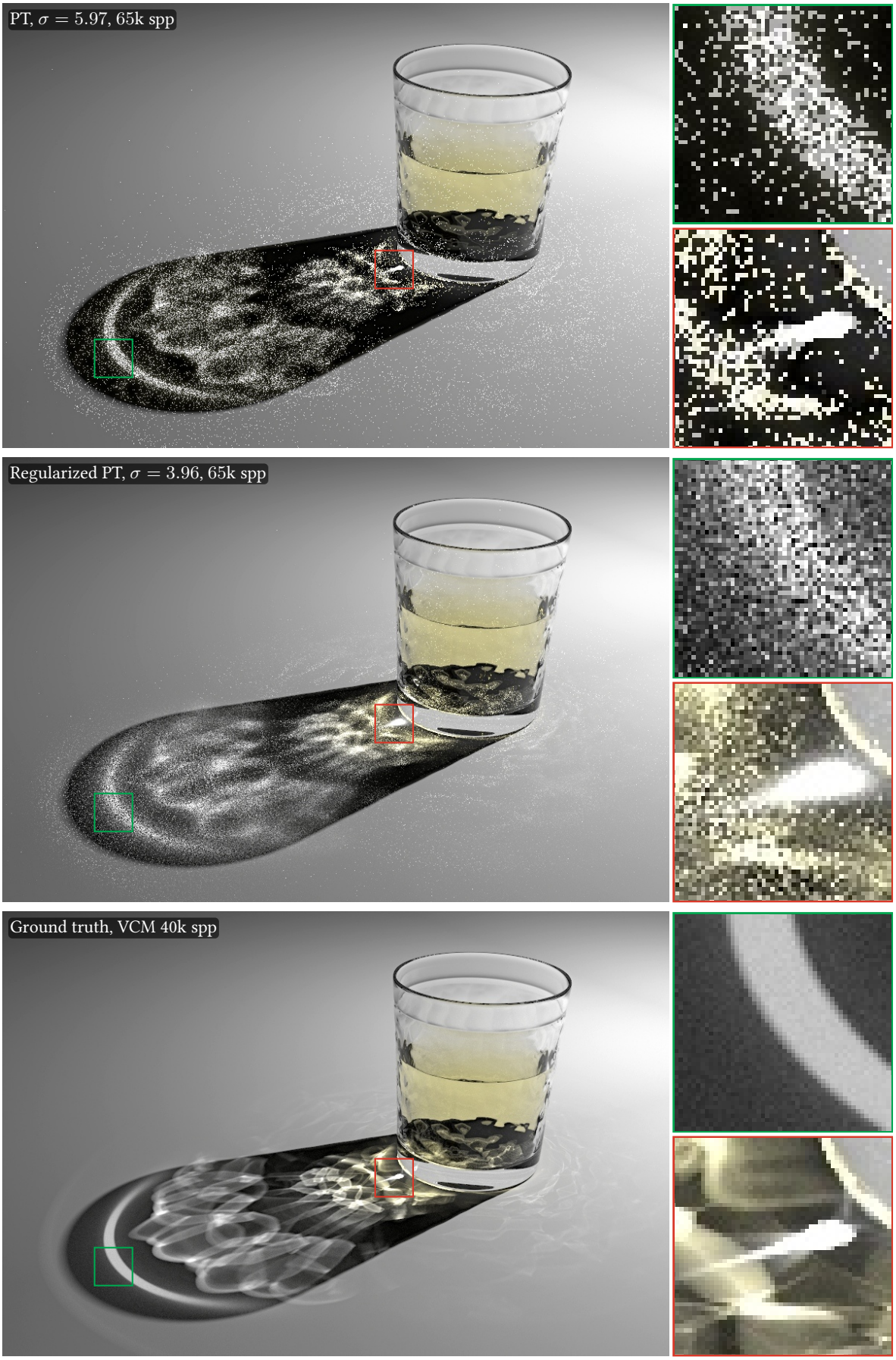
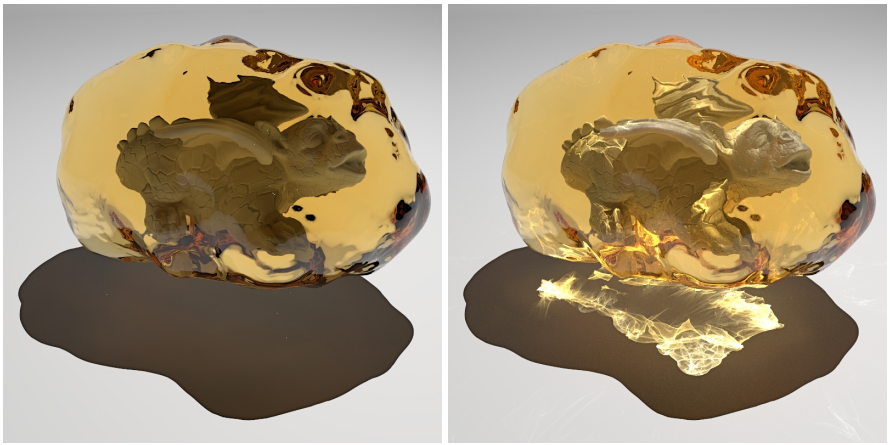
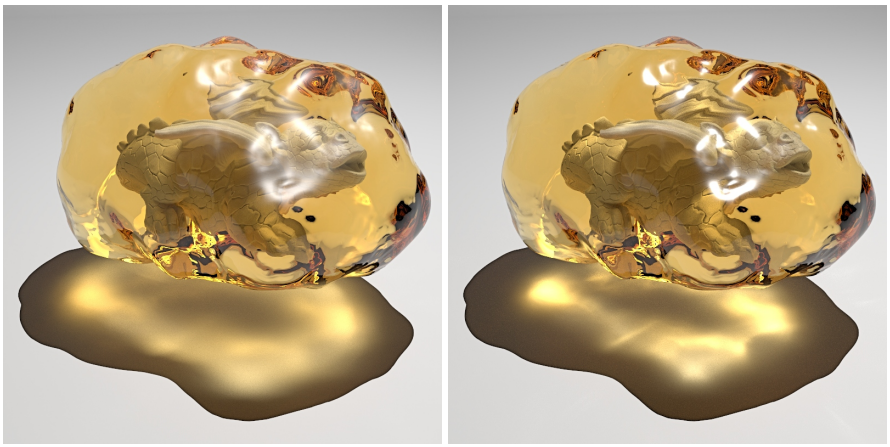


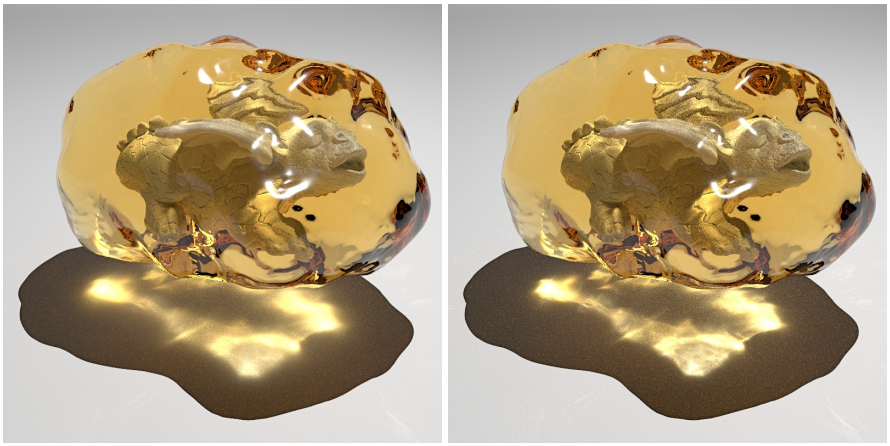
Figure V.5: Convergence of roughness-based regularization ($\tau_0 = 200$). Illumination comes from a small area light and should theoretically converge with any rendering algorithm.



(a) PT, 16k spp (b) Reference: VCM, 32k spp



(c) PT, $\tau = 10$, 16k spp (d) PT, $\tau = 100$, 24k spp



(e) PT, $\tau_0 = 30$, 32k spp (f) PT, $\tau_0 = 100$, 32k spp

Figure V.6: Bias comparison of non-consistent, regularized PT (c,d) with the consistent parametrization (e,f). The scene shows the PBRT-dragon inside an amber, illuminated by a point light source. The bias in (c) and (d) stays the same independent of the sample count (less samples were used because the variance is already acceptable).

V.5 BIAS REDUCTION HEURISTICS

In the previous figures, we can see that regularization successfully introduces light effects like caustics or SDS paths which would be missing otherwise. It reduces variance as expected, but the visible bias is large. Especially in Figure V.6 (c-f) we could have rendered the highlight much sharper (using a large τ) without risking more noise.

In this section two heuristics are proposed with two different goals. The first disables regularization in the presence of one sufficient non-regularized sampler for a path and the second scales τ to reduce the bias where possible.

V.5.1 SAMPLER QUALITY

To assess if there is a sampler with a sufficiently small variance we need to guess the variance of all samplers on the path. Regarding the observations in Section V.1 we can focus on the term C of non-sampled events. We want to restrict the standard deviation of these events to define the sampler quality.

Using our bounds, the maximum value of a connection is $M_\mathrm{c} = \bar{\rho}_k \bar{\rho}_{k+1} / d^2$. Together with the minimum value $m_\mathrm{c} = 0$, the variance bound in Equation (V.5) leads to

$$\sqrt{\mathrm{V}[C_\mathrm{c}]} \leq \frac{\bar{\rho}_k \bar{\rho}_{k+1}}{2d^2} \tag{V.27}$$

d distance between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$

as a possible guess. A standard deviation with a high absolute value is not necessarily visible, since the perceived noise depends on the relative amplitude. Therefore, I related the total path length $d_\mathrm{path} = \sum_k d_k$ to the above value. Without any scattering events, the energy would reduce with d_path^2 over the path. If the ratio d_k / d_path is very small, the variance is probably very high. The connections between the dragon's wings in Figure V.8 (a) are an example.

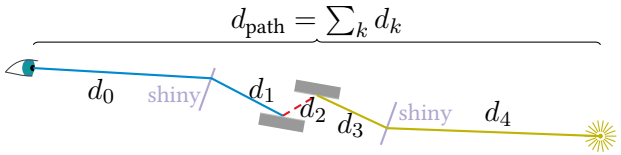


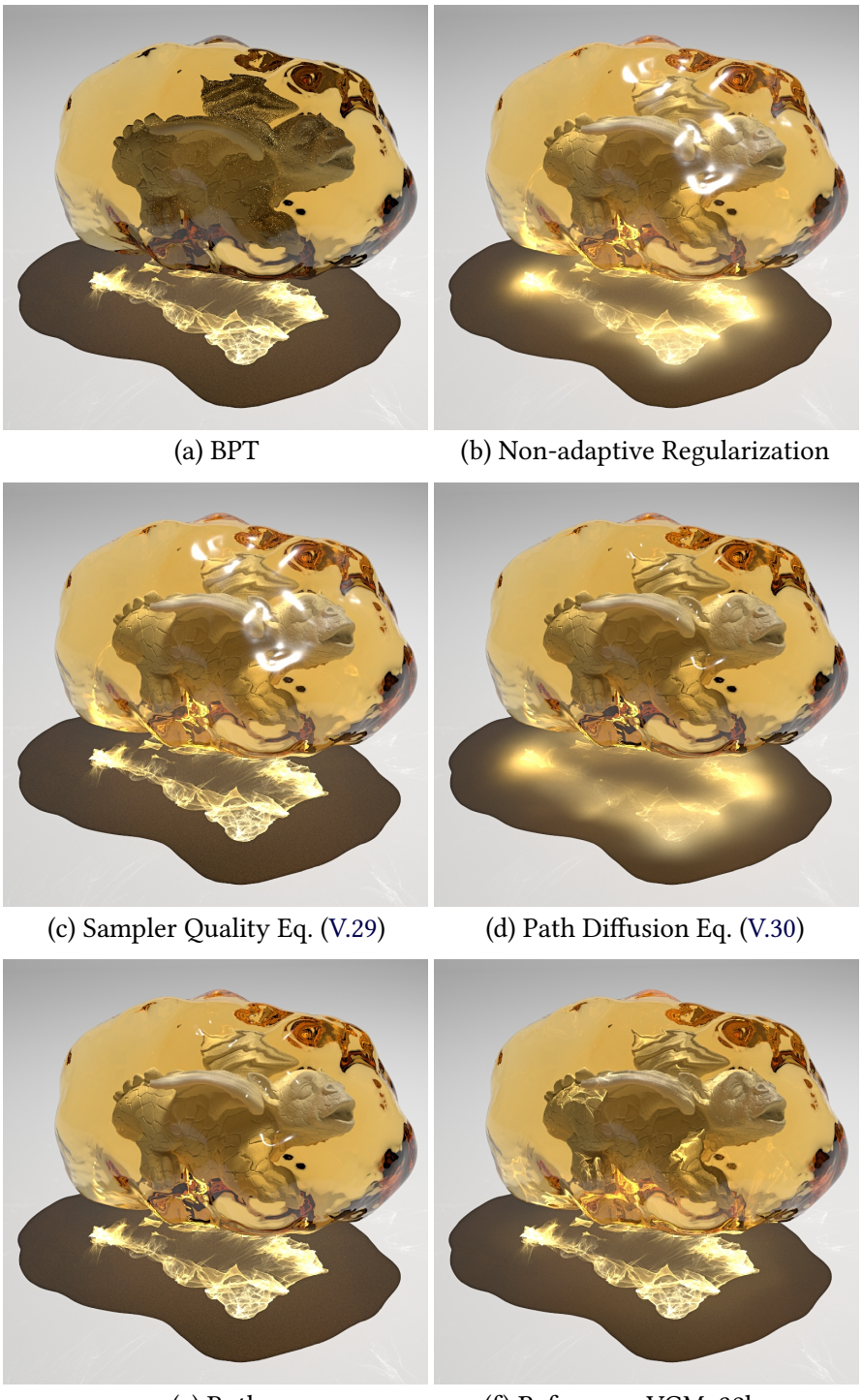
Figure V.7: Example of a sampleable, high variance path which can be avoided by regularizing other connections.

As final criterion for the best sampler's quality I ended up with

$$q = \min_k \left(\frac{\bar{\rho}_k \bar{\rho}_{k+1} d_\mathrm{path}^2}{d^2} \right). \tag{V.28}$$

If $q < \tau^2$ it is likely that at least one of the samplers has a variance which is smaller than one we could obtain through regularization. Thus, we can disable regularization by setting

$$\tau' = \begin{cases} \tau & \text{if } q \geq \tau^2 \\ \infty & \text{otherwise} \end{cases} \tag{V.29}$$



(a) BPT (b) Non-adaptive Regularization

(c) Sampler Quality Eq. (V.29) (d) Path Diffusion Eq. (V.30)

(e) Both (f) Reference: VCM, 32k spp

Figure V.8: Adaptive regularization heuristics for BPT ($\tau_0 = 30$, 16k spp). The sampler quality heuristic (c) preserves the caustic without changing the highlights. The diffusion heuristic (d) results in sharper highlights, but blurs the caustic even more. Both combined reduce the bias successfully.

Comparing Figure V.8 (b) with (c), we can see that the caustic is successfully preserved without changing regularization for the necessary paths.

Unfortunately, this heuristic has an inevitable discontinuity. If q is approximating τ^2, a small change will suddenly switch regularization on and off. This may happen, for example, if the camera is moved a small amount. Once regularization is turned on, some connections get blurred and due to their reduced variance they will immediately get large MIS weights. This effect can be reduced by using a smoothstep function instead of Equation (V.29), but is hard to eliminate completely.

V.5.2 PATH DIFFUSION

The motivation for the second heuristic is to reduce bias in visible regions. Further, for some situations, like the glossy highlights, the variance is much smaller than anticipated. Both can be improved if we scale the threshold parameter based on the diffusion of the view path, where diffusion is the variation of incident directions. At directly visible or specular reflected hits, we can see features sharply and the diffusion is small (all incident directions are similar). After a glossy or diffuse event, the details become less visible and we could apply a stronger regularization. This visibility is connected to the footprint of a path. A larger footprint area means that less details will be visible.

Footprints Sec. IV.3.3 p. 110

Until now we used τ as a limit to the angular PDF with unit sr^{-1}. Inspecting Equation (V.4), it tells us that we should try to limit $\mathrm{E}[V]^2\,\mathrm{V}[C]\,\mathrm{E}[L]^2$ to τ^2 instead. This would lead to $\sqrt{\mathrm{V}[C]} \leq \tau/(\mathrm{E}[V]\,\mathrm{E}[L]) = \tau'$ as an adaptive threshold of the entire radiance. Thereby, $\mathrm{E}[V]$ and $\mathrm{E}[L]$ can be estimated from the footprints. This means that a threshold τ' to limit the variance of $\mathrm{V}[C]$ coincides with the goal of low visibility. If the footprint area is large we are not able to see details, but we also need a stronger regularization to keep a bounded variance. Intuitively this makes sense, because a larger area will obviously increase the variation of the connection term.

Expected value from footprint Eq. (IV.11) p. 109

However, the footprint has the wrong unit m^2 and using $\tau/\mathrm{E}[V]$ to limit the BSDF would not work. Hence, it seems reasonable to use $\sigma_\triangleleft$ from the footprint estimates as our searched diffusion. While $\sigma_\triangleleft$ still has an undesired unit $\sqrt{\mathrm{sr}}$ it is at least independent of the scene scale. This invariance guarantees that the rendered image will look identical even if the scene size is changed uniformly. We divide the τ parameter

$$\tau' = \frac{\tau}{\sigma_{\triangleleft,k}} \tag{V.30}$$

by the angular deviation of the view path footprint at the respective vertex k. For the computation of $\sigma_\triangleleft$ please refer to Table IV.1.

In Figure V.8 (d) it is shown that the diffusion heuristic successfully increases the sharpness of the highlights. It also detects that, after the diffuse bounce on the floor, a much stronger regularization is required to keep a low variance for the caustic. This ignores the fact that there is another sampler which can sample the caustic well. Therefore, combining both heuristics (image (e)) produces a much better result than any of the two heuristics alone. Applying both we avoid most of the unnecessary bias. Comparing images (a) and (e) to (f) we see that the regularized BPT is much closer to the ground truth than the vanilla BPT.

V.6 EVALUATION

Path space regularization can successfully reduce variance on difficult light paths and makes specular paths (LS^+E) sampleable after all. The most prominent examples are the comparison between no regularization and our approach including the two heuristics (Figures V.1,V.8 and V.6).

While the virtual merge technique appears visually more appealing it also introduces noise on its own. Contrarily, the roughness-based approach causes a lot of blurriness to reach moderately noisy images. By hiding its most visible bias problems with the two heuristics, the *adaptive* method produces the best results and will be further tested in the following.

In addition to the average standard deviation σ the error of *relative* residuals RMS*R*E

RMSRE introduction and example Sec. III.3.1

$$\text{RMSRE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{(a_i - b_i)^2}{(a_i + b_i/2)^2}} \tag{V.31}$$

a_i, b_i pixel values of a reference and the given image with N pixes.

is annotated in Figure V.10. Using absolute RMSE penalizes bright noise much more than missing features and turned out to be unusable. For example in the TOY DRAGONS scene (Figure V.9), the PT and the VCM rendering have almost the same RMSE. The relative RMSRE is symmetric and rates the errors in a more meaningful way.

Figures V.9, V.10 and V.11 show renderings of PT, BPT and VCM each with and without regularization. The scenes are chosen to have many difficult light paths. For simpler scenes, regularization would be adaptively disabled for most paths and therefore would only decrease rendering performance.

A first thing to notice is that all methods produce very similar results with regularization turned on. Without it, PT and BPT are often lacking caustics and SDS paths. However, even with regularization, the quality of the respective effects differs. Regularization only turns non-sampleable paths into sampleable ones, while using a better sampling method may produce these paths with much higher likelihood. This is especially visible in the sharpness and noisiness of the caustics in the TOY DRAGONS and the MARBLES scene.

Surprisingly, the variance measured by σ increases in all scenarios when regularization is turned on. This happens because the otherwise missing light paths are introduced with a high variance. Comparing the RMSRE we can see that the combined error of variance and bias is decreasing everywhere. This is in accordance with the observation of a much higher similarity between the regularized images.

If we take a closer look at VCM, we can also observe areas of reduced variance: see Figure V.9 third closeup, Figure V.10 second and third closeup and Figure V.11 first closeup. In the TOY DRAGONS scene this causes a loss of detail in the shadow on the wing. The reason is that our sampler quality heuristic does not include the merges, which are the only low variance samplers for these paths. If we respect the merges, regularization would be adaptively disabled in this region yielding the original output. However, since merges are a kind of regularization as well, it is also logical to treat them as such. A solution to compensate the loss of detail in SDS paths would simply be to increase the threshold τ_0 if applied in a VCM-like method.

Sample Quality Sec. V.5.1 p. 147

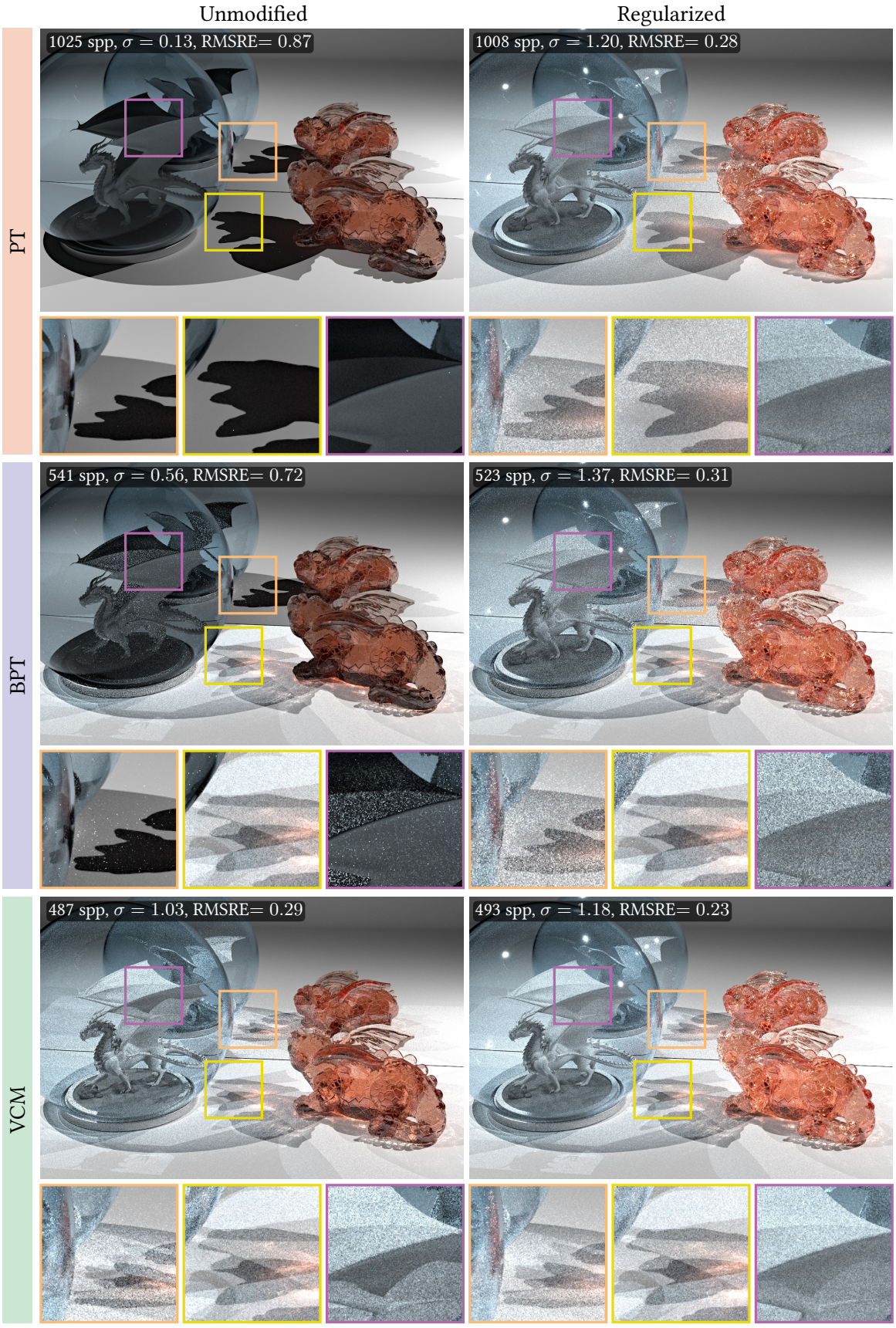


Figure V.9: Equal-time comparison (1 h) of a scene with two TOY DRAGONS in front of a mirror, one dragon inside a thin glass hull; 2 point lights. Adaptive, consistent ($\tau_0 = 10$) regularization on the right.

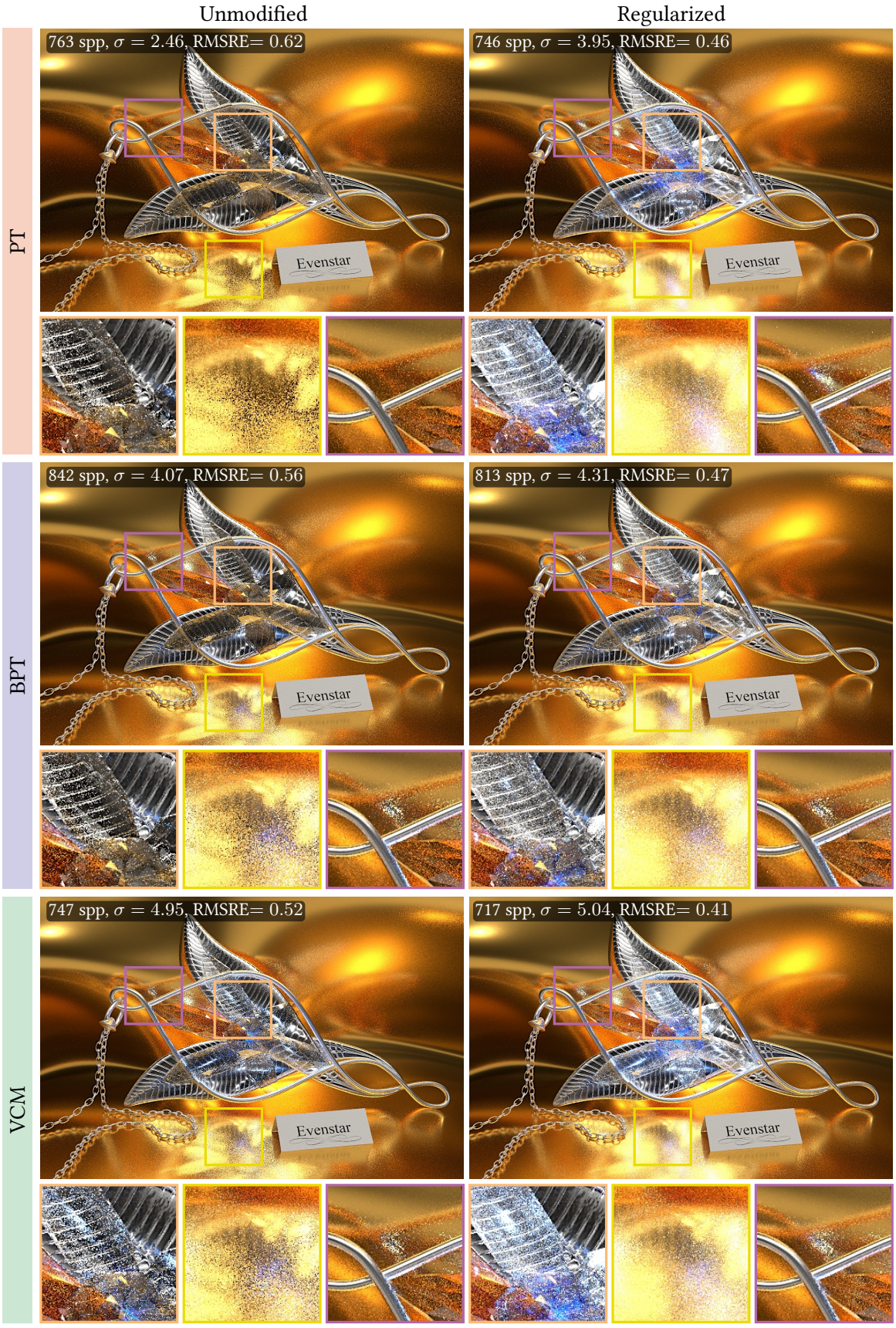


Figure V.10: Equal-time comparison (1 h) of a NECKLACE with several crystals on a roughened golden floor; 10 point lights (7 inside the crystals), 1 area light. Adaptive, consistent ($\tau_0 = 10$) regularization on the right.

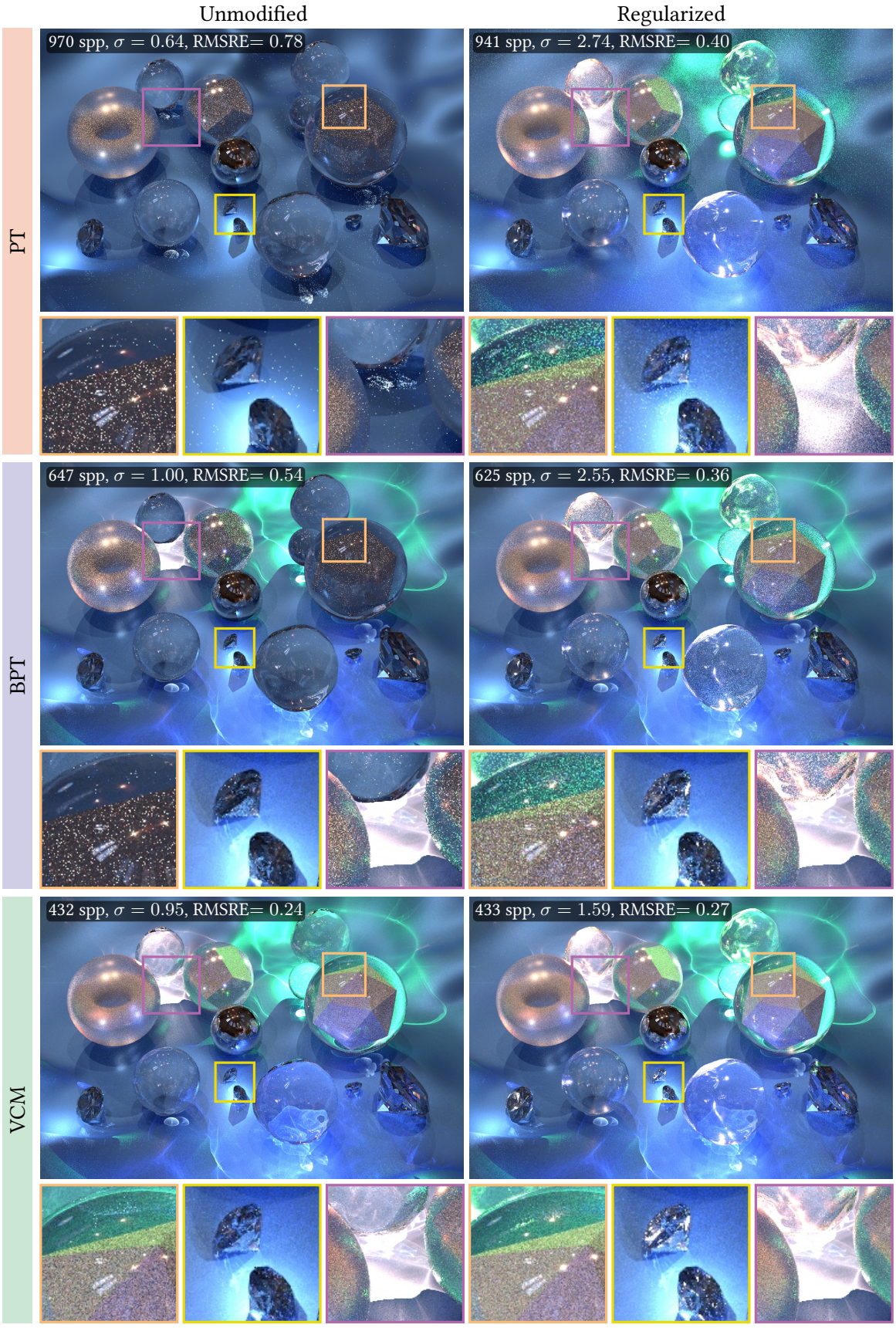


Figure V.11: Equal-time comparison (1 h) of the MARBLES scene: glass / metal spheres and gems on a blue cloth; 6 point lights (3 inside spheres), env.-map. Adaptive, consistent ($\tau_0 = 10$) regularization on the right.

Considering the number of iterations in the equal-time renderings, regularization has an overhead of roughly 3.3%. The main source of overhead is the use of the adaptive heuristics. Since parts of the heuristics depend on information which is only available after a path is completed, values along the path must be reevaluated for each successful connection or merge event. Without these heuristics there is no measurable performance loss.

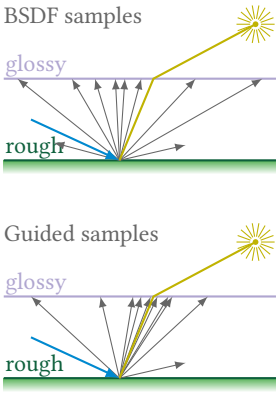
V.7 DISCUSSION: FUTURE WORK

In general, it is possible to render more difficult paths with simpler sampling methods using regularization. This is an advantage for production renderes which are based on PT. By adding this simple modification they can produce results closer to a more complex algorithm.

However, regularization will not produce an unbiased and low noise result in a practical amount of time, if the underlying algorithm cannot handle the specific paths. It only enables otherwise not sampleable paths, without increasing their likelihood much. From my perspective it is better to use the more expensive VCM instead of a PT or BPT with regularization in the current form. Enabling regularization in a VCM still adds the glossy paths and partially reduces the variance, so it is an improvement to the robustness.

A first avenue of future research is the fundamental regularization operator. In this chapter I compared virtual merges and the roughness-based approach of which the first produces shaper highlights at the cost of more variance. In Section V.2.4 a first alternative is discussed. Another issue to consider is the distance and the cosines from the connection term. As stated before, any BSDF focused regularization strategy will not be able to compensate the variance caused by the other terms.

Variance discussion V.1
p. 131



To increase the effectiveness of regularization, a combination with a general guidance framework [Herholz et al. 2016; Müller et al. 2017] seems to be very promising. The idea of guidance is to steer the local samplers into the direction of the incoming adjoint quantity. This by itself is not sufficient to find glossy paths, but is very likely to increase the number of useful regularized connections. While regularization increases the size of highlights, guidance increases the likelihood to move into the direction of the highlight.

A different application where regularization could work quite well is the use in improved BPT methods. Popov et al. [2015] proposed to resample a connection attempt from a set of possible connections with respect to the expected contribution. Like guidance this increases the chance to select good connections for regularization. Similarly, Chaitanya et al. [2018]'s *Matrix Bidirectional Path Tracing* reshuffles a set of connections to minimize the connection distances inside this set. This again, increases the expected contribution and reduces the variance of connections in general.

Finally, we can improve the heuristics to reduce the bias. The most interesting solution would be to find a general MIS weight framework, which minimizes bias and variance at the same time. Such a framework would behave similar to the sampler quality heuristic, but would hopefully be more reliable.

To enhance the path diffusion heuristic it is also possible to try different footprint approximations. Therefore, the highest quality candidate would be the *Covariance Matrix Tracing* [Belcour et al. 2013].

Another problem is the temporal incoherence of the sampler quality heuristic. It can be reduced by using some smooth transition instead of a hard threshold, or by using an estimate without dependence on the path length. All other parts of the approach are already temporal coherent.

Chapter VI
Next Event Backtracking

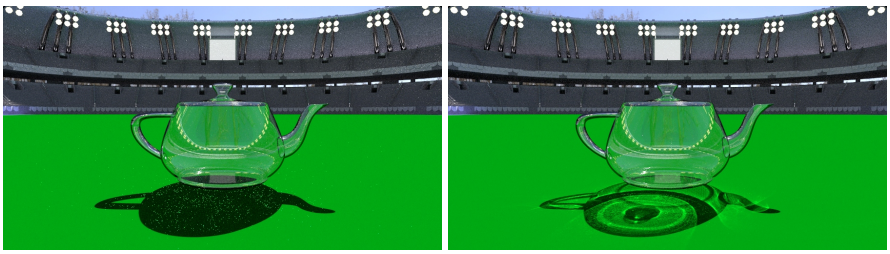


Figure VI.1: Equal time comparison (15 min) of BPT (left, 212 spp) and PT with Next Event Backtracking (right, 341 spp).

Besides the material there are even more reasons for a sampler to become infeasible in practice. As discussed in Section V.5.1 the relative length of a segment opposed to the entire path length is an important criterion. Bridging large distances with random walks produces smaller densities (= larger footprints). If the size of the footprint is large relative to the merge radius or the length of the next connection segment, this causes a high variance. On paths with rough vertices this is no problem because the long segment can be captured by a connection. However, if the end vertices of the long segments are specular or glossy, other samplers are enforced.

A very famous example is the teapot in a stadium (Figure VI.1). The small glass teapot throws a caustic, but the photon density in the region of interest is far too small. BPT and all photon related methods fail to find the caustic in this scenario without guidance.

The publication [Jendersie 2019a] got rejected on EGSR 2019. After incorporating the improvements from the reviews I put a version on arXiv for reference in this thesis:

Next Event Backtracking
Johannes Jendersie
In: arXiv:1909.00573

The basic idea of *Next Event Backtracking* (NEB) is to use the segment of a next event estimation as the first segment of a photon path. Thus, the photon transport inherits the strengths from NEE, including the guidance for many lights. Thereby, photons are guided toward regions of high importance without learning or sampling an explicit guide.

NEE Sec. II.7.4 p. 67

This new operation requires to convert the differential irradiance at the point of NEE into a flux. Therefore, we need to know the density per area of NEEs which are performed at the respective position. This motivated me to develop the density octree from Section III.2 p. 89 to obtain fast and high precision estimates of the required density.

Differential Irradiance Eq. (II.5) p. 16
Flux Eq. (II.2) p. 15

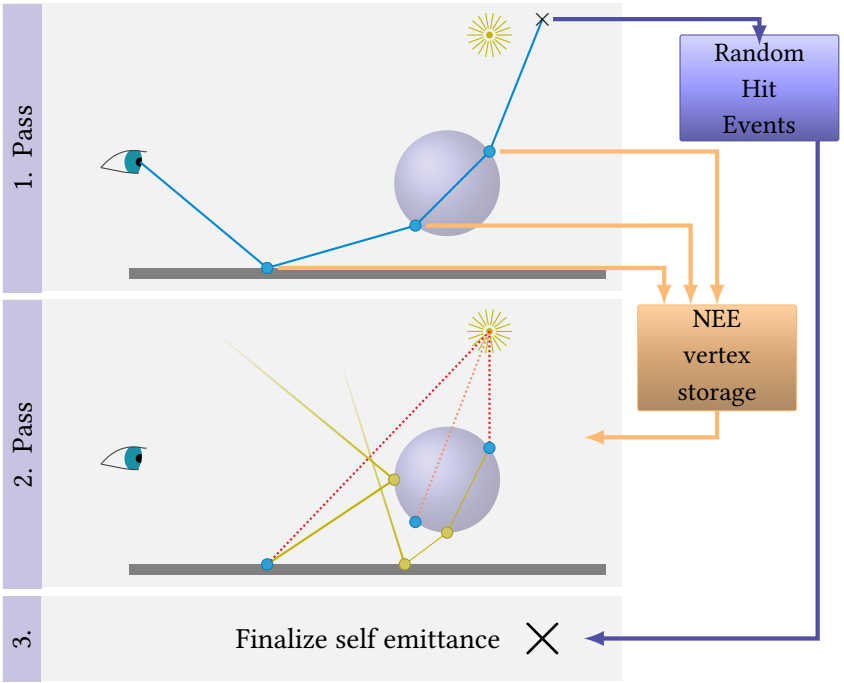


Figure VI.2: Algorithm outline: In the first pass, a PT is executed and the intermediate results are stored. In the second pass, one photon is traced from each of the non-zero-estimate vertices from pass one. Finally the contributions from random hits, photons and NEE are weighted to compute the final contribution.

VI.1 THE ALGORITHM

The basic idea of using a connection as a segment of a photon path is not limited to the algorithm in the following. However, it is reasonable to explore the strength of the NEB operation starting with a PT as underlying method. The outline of the algorithm (Figure VI.2) is as follows:

Path Tracing (PT) Sec. II.8.3 p. 73

1. Trace paths as in *Path Tracing*
   (a) Store the hit-points (called *NEE vertex*)
2. NEE and photon tracing
   (a) Estimate NEE vertex density
   (b) Compute NEEs
   (c) Trace photons and apply contribution directly
   (d) [Optional] Trace photons from the light source as usual
3. Compute self emittance contributions

VI.1.1 TRACE VIEW PATHS

For tracing paths we use a conventional Path Tracer. However, it is not possible to compute the results for random hits and NEE immediately. To calculate the MIS weights, it is necessary to know the photon events which themselves depend on the density of NEE vertices (which are being created in this pass). For this reason the intermediate results (hit point, incident direction, originating pixel, throughput and relative path PDFs) for the events must be stored.

VI.1.2 NEE AND PHOTON TRACING

Before it is possible to trace a photon, the differential irradiance at the NEE vertex (unit $\mathrm{W\,m^{-2}}$) must be converted into a flux Φ (unit W). This is achievable by multiplying the incoming irradiance with the differential area $\mathrm{d}A$:

$$\Phi = \mathrm{d}E \cdot \mathrm{d}A = \frac{\mathrm{d}E}{D} \tag{VI.1}$$

where D is the local density of vertices. By doing so, we interpret the NEE vertex as a virtual light source with an area $\mathrm{d}A$. We can insert $\mathrm{d}A = 1/D$ because the area, which is associated with each vertex, depends on this density: If there are more vertices, each one represents a smaller part of the surface area.

Density D Eq. (III.7) p. 88 and Eq. (III.8) p. 92

Note that it is not important how the vertices were generated. It only matters how many vertices are stored in a local area to turn each of them into an unbiased[1] emitter. Especially, the past (sampling events, Russian roulette, ...) of the view path vertices is not important.

Russian roulette Sec. II.7.3 p. 65

For the estimation of the particle density in step (2.a), k-nearest neighbor searches or one of the data structures from Section III p. 83 can be used. As all density estimates are biased, NEB is a biased estimator too. Unfortunately, we cannot make this estimate consistent, because in each iteration

[1] Unbiased, if we know the true density, which is not the case

we need new queries at the new locations. Therefore, using the octree, which converges over time, results in the least total error.

Now that the density D is known, the NEE contribution can be finalized (step (2.b)). This was not possible before, due to the unknown MIS weights. The computation of the weights themselves is detailed in Section VI.2.

Next, in step (2.c), photons are traced by beginning a new random walk at the NEE vertex. The first sampling event is based on the NEE connection direction as incident direction and the vertex's BSDF. It is not necessary to store photons, because we can invert the search for merge events. Instead of searching for photons around the view path vertices, we can as well search for view path vertices around photons, which produces identical results. Therefore, the NEE vertex storage must be some kind of spatial data structure to support neighborhood searches.

Optionally, it is possible to trace photons from the light source (step (2.d)) and add their contributions in the same way as the photons from NEE vertices. Only the MIS weights must be adapted accordingly. I will demonstrate in section VI.4 that adding those photons complements NEB where it is weakest.

VI.1.3 COMPUTE SELF EMITTANCE CONTRIBUTIONS

Since it was not possible to compute the MIS weights properly before step (2.a), we needed to store random hits of the light source in pass 1. Now, we can iterate over the stored events and compute the contributions.

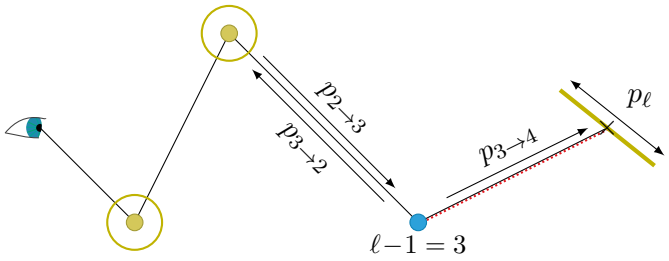


Figure VI.3: MIS-weight computation between several events along the same path. The shown example path has length $\ell = 4$.

VI.2 MIS Weights

In the basic algorithm there are three types of events which must be weighted against each other. When conventional photons are traced too, there is one additional event type. Each path of length ℓ has a single random hit event, one NEE event if $\ell \geq 2$ and $\max(0, \ell - 2)$ photon merge events as depicted in Figure VI.3. Optionally, there can be $\max(0, \ell - 1)$ merges with usual photons.

To compute the balance or power heuristic, the path PDFs and the reuse counts are required. We have one random hit event ($N_{\text{rv}} = 1$) and one NEE event ($N_{\text{c}} = 1$) for which the PDFs from Sections II.7.3 and II.7.4 are repeated here:

Power heuristic Eq. (II.24) p. 23

$$p_{\text{rv}} = p(\boldsymbol{x}_0) \cdot \prod_{i=0}^{\ell-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)$$

p_{rv} Eq. (II.77) p. 66

$$p_{\text{c}} = p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{\ell-2} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot p(\boldsymbol{x}_\ell).$$

p_{c} Eq. (II.80) p. 67 with $k = \ell - 1$

The PDF for the new operator is closely related to that of NEEs. Each photon path starts with a next event estimation and therefore has to include $p(\boldsymbol{x}_\ell)$. From there, a random walk in backward direction is performed up to the merge vertex $\boldsymbol{x}_k$. Further, it consists of another random walk beginning at the observer. As the last ingredients, we need the probability for a successful merge $D_{\ell-1} \cdot \pi r^2$ and the total number of photon path starting points N_T, i.e. the number of stored NEE vertices. Together this gives

$$p_{\text{neb},k} = p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{k-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot \frac{D_{\ell-1} \cdot \pi r^2}{N_T} \cdot \left[\prod_{i=k}^{\ell-2} p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1})\right] \cdot p(\boldsymbol{x}_\ell) \tag{VI.2}$$

and $N_{\text{neb}} = N_T$ for the sampler count because we reuse photons from all paths. Note that for a more robust solution we would need to compute a better reuse factor as described in Chapter IV p. 101.

The above merge probability is derived as follows: First, we need the probability density per area to start the respective photon random walk. This is the density of NEE vertices at the second last vertex $D_{\ell-1}$ which we computed in step (2.a). This density must be divided by the particle count N_T, since the density octree itself returns the number of events per area, but we need the PDF per area of a single sample. Next, the other

probability densities $p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1})$ to get from vertex $\ell-1$ to the current vertex are collected in the second product term. Finally, the merge probability must be proportional to the size of our search region πr^2. The larger the search region, the larger the chance to find something. The three values together give the chance to create and find the respective NEB photon path.

Finally, we have

$$p_{\mathrm{m},k} = p(\boldsymbol{x}_0) \cdot \left[\prod_{i=0}^{k-1} p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)\right] \cdot \pi r^2 \cdot \left[\prod_{i=k'+1}^{\ell} p(\boldsymbol{x}_{i-1}|\boldsymbol{x}_i)\right] \cdot p(\boldsymbol{x}_\ell) \qquad p_{\mathrm{m}} \text{ Eq. (II.83) p. 68}$$

for the conventional photons. The number of samples is N_Φ due to global reuse of photons or the improved estimate from Chapter IV p. 101 again.

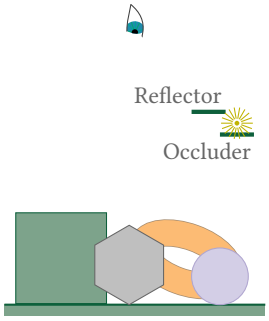
Plugging the above path densities in the balance heuristic, we can compute the MIS weights recursively along the path as usual.

VI.3 Results

The next event backtracking operator has clear strengths and weaknesses. It is strong whenever NEE is more likely than other events on the path. This is the case for small or distant light sources like in Figure VI.1. Additionally, it scales well with many lights, if contribution-sensitive NEEs are used. It is weak whenever the vertex density is much smaller than the light contribution, which is the case for low visibility regions or if the light source is close to a receiver surface.

Figure VI.4 shows an equal time comparison of the NEB method to other rendering algorithms in selected light situations. For the first two scenarios our method is superior due to the uniform sampling density of NEE vertices on the glass surfaces. The Mirrors scenario shows a small degeneration in quality where the distance to the caustic receiving surface increases (mirrors at the top). The Reflector scenario represents the worst case. Here, the tiny, bright surfaces close to the point light source are rarely found by a Path Tracer.

Reflector scenario



In Figure VI.5 more realistic scenes are compared. For situations like in the Watch scene, NEB is superior to the state of the art VCM. In other situations (Bathroom or Christmas scene) it shows more noise than VCM without modifications. Enabling the additional tracing of light photons makes NEB equally effective as VCM. In all my experiments I found that NEB+LP is very strong for caustics and SDS paths regardless of the scene scale. For diffuse indirect lighting it performs on a comparable level to most other methods.

NEB+LP means plus conventional *light photons*, option (2.d)

VI.3.1 Bias

In Figure VI.6 a scene with many separated lighting situations can be seen. It demonstrates the relatively small bias which is introduced by the density estimates. While there is a measurable bias, it is barely visible. Even in the most severe cases, a good display device is necessary to see that NEB is slightly brighter in some regions (closeups). The worst observable error happens if the planarity assumption in the octree's density calculation does not hold. In Figure VI.4 (Mirrors) one such artifact can be found on the edge of the topmost caustic. A possible solution would be to additionally split the octree dependent on a local curvature criterion, to ensure that the planarity assumption does not fail catastrophically.

Octree density Eq. (III.8) p. 92

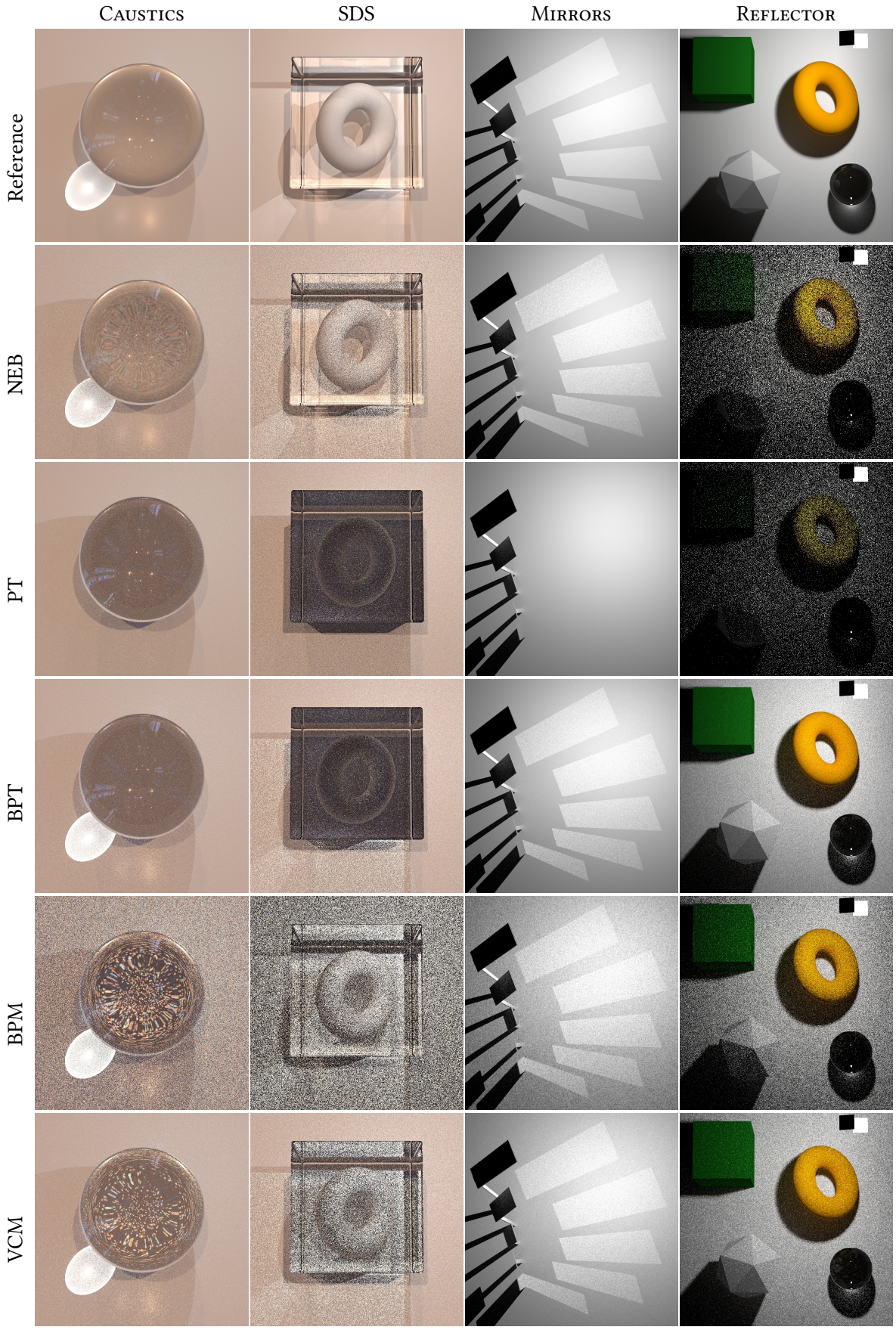


Figure VI.4: Equal time comparison (1 min) of different rendering algorithms in difficult light scenarios.

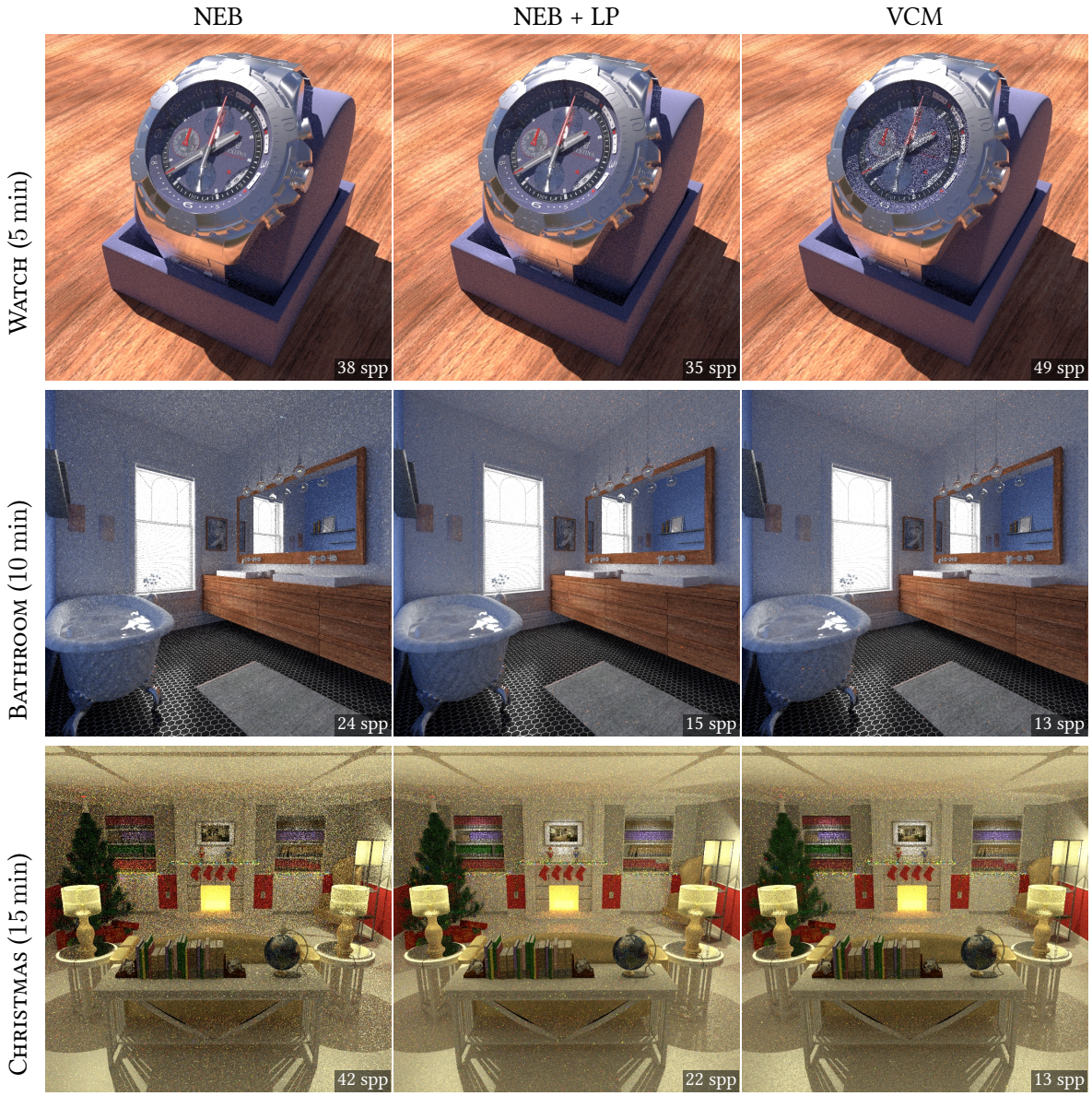


Figure VI.5: Comparison of NEB to VCM in more realistic scenes.

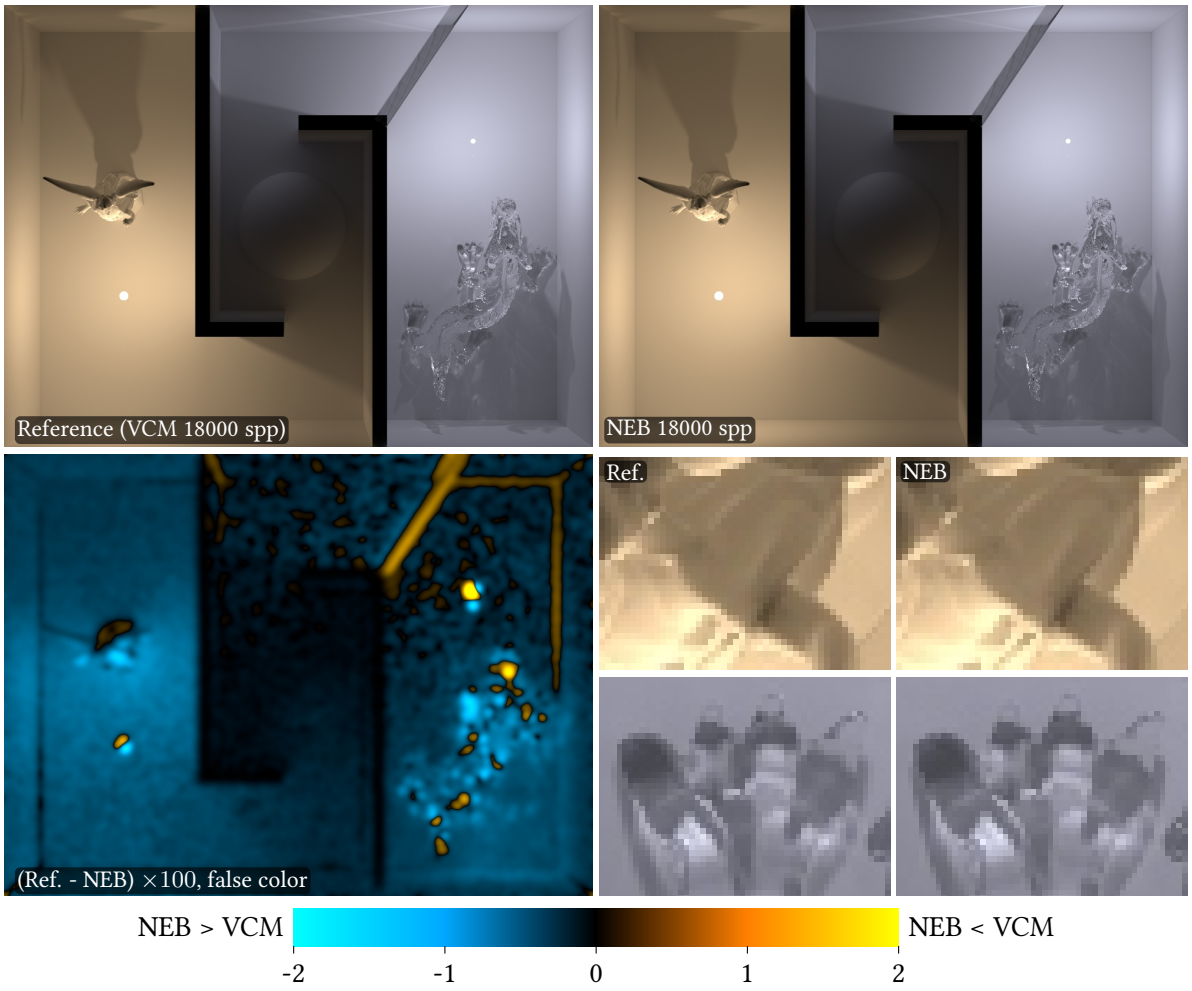
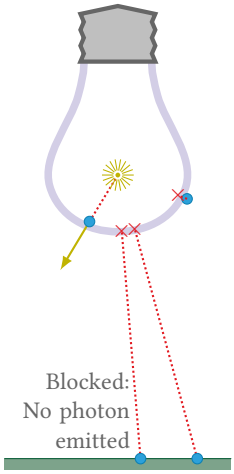


Figure VI.6: Bias visualization. The scene shows multiple bounces of indirect lighting, caustics and SDS paths. The bottom left image is the blurred difference image times 100 in false colors. Without the blur we would only see signed noise, whereas the blurred version shows systematic deviations (areas of a single color) versus unbiased, noisy errors ("wobbly" pattern). For most parts of the scene, NEB yields the systematically larger values (blue regions). But, even in the regions with the largest error, the bias is barely visible (closeups).

VI.4 MODIFICATIONS

NEB as a fundamental operator allows many modifications of the algorithm itself. The PT-based algorithm from Section VI.1 is by no means the only possible realization. In this section further changes to improve the robustness of the method are proposed.

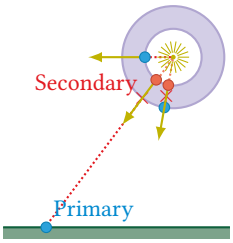
VI.4.1 CONVENTIONAL LIGHT PHOTONS

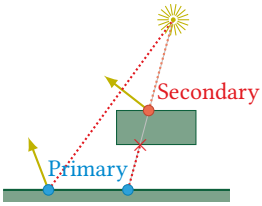


While the basic algorithm is very strong in scenarios like the teapot in a stadium (Figure VI.1), it fails when the caustic throwing object is much closer to the light source than to the receiver. Consider the example of a light bulb – an emitter inside a glass ball. Only NEEs on the inner surface of the bulb produce contributing photons, but only very few paths randomly hit the comparable small bulb. All other vertices in the scenes have no NEE contribution, because the glass ball is blocking the connection to the light source. Therefore, the number of useful NEEs and photons are both very small, leading to high variance results.

We observed that the failure cases of NEB occur in situations where the conventional photon tracing, which starts at light sources, is strong. Hence, combining NEE photons and light photons (LP) by the means of MIS promises a more robust algorithm. In Figure VI.7 the effectiveness of this combination is demonstrated. The renderer becomes much more effective with respect to time although its iteration count decreases. The reason for both is that many more photons are found and merged at each position. Besides the additional tracing of photons, this requires more evaluations of the BSDF and the MIS weights.

VI.4.2 SECONDARY NEES

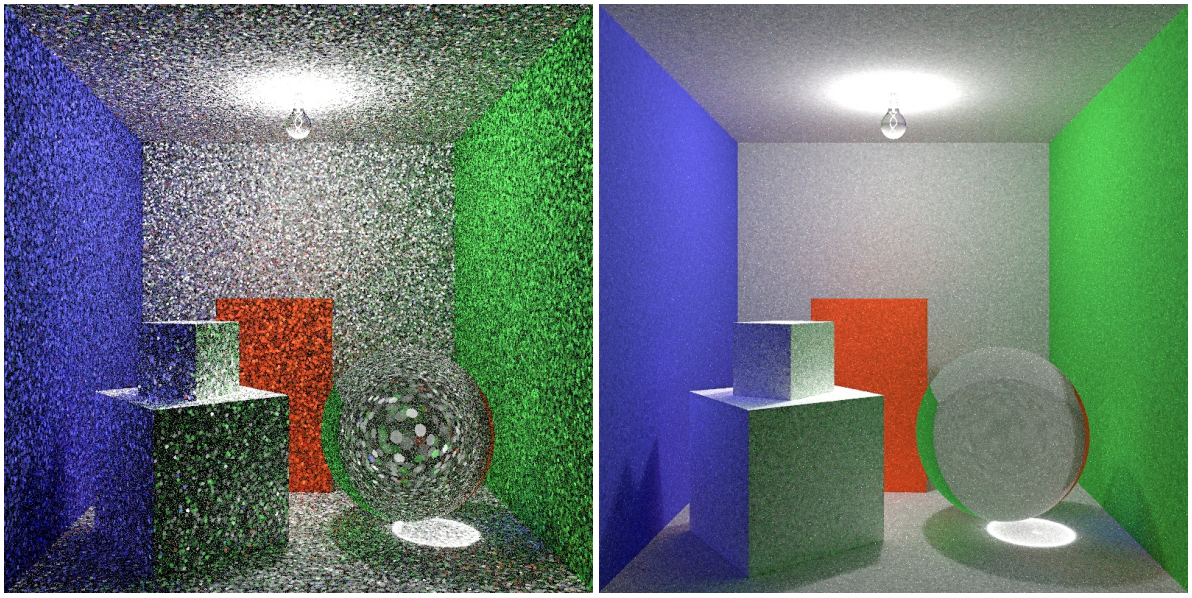




Another option to solve the light bulb scenario is to use the intersection with the blocker closest to a light as emitter, if a NEE shadow test reports an intersection. Right now only NEEs on the bulb surface itself generate contributions, but there will be many more blocked shadow tests.

The difficulty is to turn the intersection point into a virtual light emitter. As before, the density of these intersection points must be known for which the new points must be inserted into the density data structure, too. Additionally, we need an unbiased estimate of the differential irradiance for the intersection point. It is not possible to directly use or rescale the value from the primary vertex. Instead, a new NEE for the new point must be performed. These can be shadowed again if a connection to a different light source is selected. Hence, the name secondary NEE.

Figure VI.8 uses the same setup as Figure VI.7 but shows the results of secondary NEEs instead of light photons. The method has a higher variance than using normal light photons due to the chance of shadowed or badly sample secondary NEEs.



NEB, 68 spp NEB + LP, 14 spp

Figure VI.7: Equal time comparison (5 min) without and with Option 2.d in a light bulb scenario.

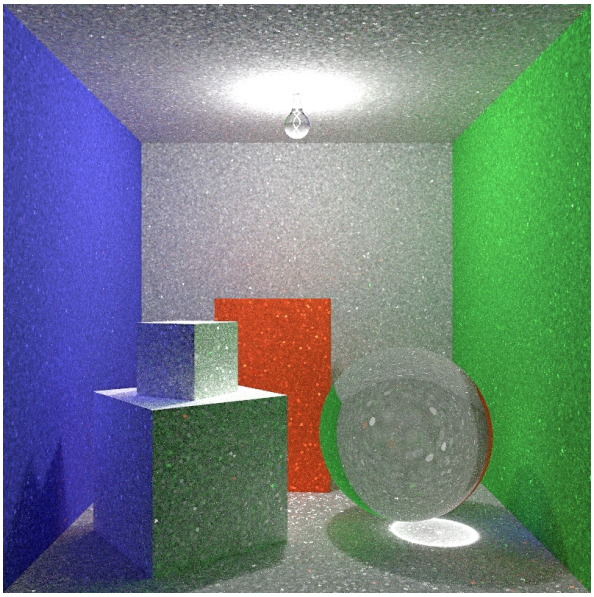


Figure VI.8: Equal time result (5 min, 21 spp) for the scene from Figure VI.7 using the secondary NEE modification.

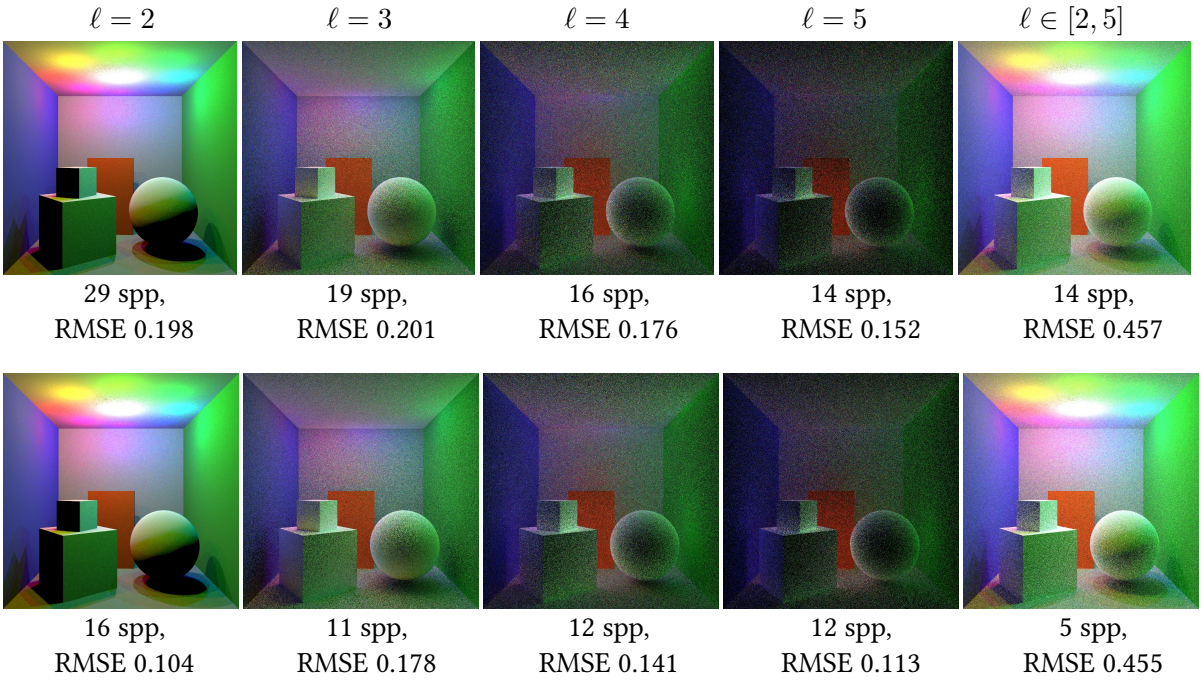


Figure VI.9: Equal time comparison (1 min) of Path Tracing fixed to path length ℓ without (top) and with (bottom) NEE recycling.

VI.4.3 Discussion: Splitting

One reason for the bright splotches in NEB is that there are NEEs with a very high contribution (short connection) but a low vertex density. This leads to very bright photons being emitted. Examples for this situation are the light bulb and the tiny reflector scenarios. If the vertex density were as large as the irradiance of a surface, NEB would perform significantly better.

An option to increase the number of emitted photons is to split the flux into smaller packets and to trace multiple paths. This would work quite well for the tiny reflector, but not at all for the light bulb. On a specular surface all emitted photons would start into the same direction. Thus the splitting would be completely useless.

A method which could work in both cases is to split the vertex and to distribute copies on the local manifold. Unfortunately, I did not find time to explore this idea further.

VI.4.4 NEE Recycling

Since we already store the NEE vertices in a search data structure, it seems reasonable to share the results of NEE events. Using arbitrary types of range queries allows to find neighbored NEE vertices. Then, all available NEEs at one vertex can be averaged and the effective count of NEEs N_c for the MIS weights increases to the number of found events.

In Figure VI.9 an experiment with and without NEE reuse is shown to judge the effectiveness of the proposed modification. Enabling the merges is clearly slower due to the range query and the additional evaluations of BSDFs. Despite the lower number of samples, the noise level (*Root Mean Squared Error*) is slightly better when reusing the NEEs. However, the ef-

fectiveness decreases with path length and gets worse than usual PT for practical path lengths. Repeated experiments with different merge radii had the same outcome. The reason for the low effectiveness is that the noise in indirect lighting is dominated by the random walk and not the NEE.

Concluding, the idea of reusing the NEE events sounds promising, but does not pay off in this form. Therefore, I used only the one primary NEE without this modification in all other experiments.

VI.5 Comparison to Other Methods

The method which behaves most similar to NEB is *Manifold Next Event Estimation* (MNEE) [Hanika et al. 2015a]. If a shadow ray of a regular NEE is blocked by a refracting surface, MNEE iteratively moves the intersection vertex/vertices until the specular contribution path is found. This makes it possible to find caustics and SDS paths in a PT setting. MNEE is relatively expensive, since each new connection in the iterative process must be checked for occlusions again. Also, it might have a bias if there is an ambiguity or if the caustic throwing object is not a blocker. Due to the deterministic process ambiguities are not explored completely.

The weakness shared between NEB and MNEE is that not all caustics will be found with equal likelihood. MNEE can only detect caustics from blocking surfaces while NEB has to randomly hit the respective surface. A major difference is that in NEB photons are generated and merged which increases the effectiveness at the cost of more memory. The original intent behind NEB was to find an alternative to MNEE. However, besides lower efficiency, using a single backtracking path is even more complex than sharing the results between all NEEs. For the conversion of flux we would need the density of NEEs with respect to the generating path. This needs the footprint of the path, of which we have only strongly biased estimates.

Footprints Sec. IV.3.3 p. 110

Comparing NEB as a method to produce high quality photons maps against the older methods [Keller and Wald 2000; Peter and Pietrek 1998; Suykens and Willems 2000], it produces a medium quality map. In most scenes, it performs similarly to the other methods without wasting samples for the estimation of importance distributions. However, if there are small reflectors close to the light source (light bulb or Figure VI.4) it will be less effective, because it does not respect the radiance distribution. Adding standard photons compensates this weakness, but their distribution does not follow importance again. However, since none of the photons is ever stored, the memory consumption of additional photons is of no interest.

Chapter VII
Pixel Cache Light Tracing

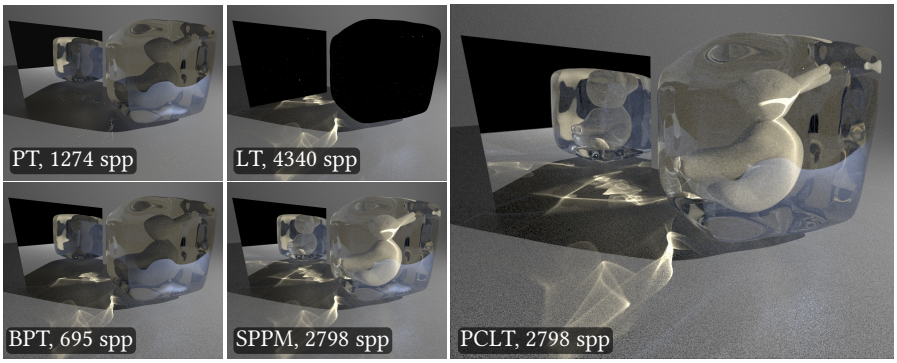


Figure VII.1: Equal time comparison (1min) of different sampling strategies. The surfaces are not perfectly specular and any of the methods should converge, which will not happen in practice.

The last method I want to include in this thesis is one of my first publications, *Pixel Cache Light Tracing* (PCLT) [Jendersie et al. 2017]:

Pixel Cache Light Tracing
Johannes Jendersie, Kai Rohmer, Felix Brüll, and Thorsten Grosch
In: Proc. of Vision, Modeling and Visualization, pp. 137–144

The overall goal was to build a GPU specialized rendering method which has low divergence and low memory consumption. Other than CPUs, GPU cores do not process data independently. Instead so-called warps (groups of 16 or 32 threads) share one instruction unit and are deemed to process the same steps as all other cores of the same warp. Hence, GPUs are good at performing the same, preferably small, task many times in parallel. Divergence occurs if different threads take different branches of the program. Then all other threads are masked out and computation time is wasted.

A consequence of the GPU architecture is that most of the Monte Carlo methods do not scale well if ported naïvely. Davidovič et. al explored several approaches to perform BPT and VCM on a GPU in his survey [Davidovič et al. 2014]. However, even using single stochastic connections in BPT instead of the full connection to all vertices of a light path makes BPT a bad choice. As can be seen in Figure VII.1, (stochastic) BPT only achieves a quarter of the iterations of SPPM in the same time. This is because SPPM never attempts a connection or shadow test and thus needs less divergent ray

tracing events.

A method to avoid divergence due to different path lengths is the regenerative approach, where a thread fetches a new path after completing another one. This improves the utilization (less idle time), but cannot fix the divergence problem of dynamic connection counts in BPT.

A general possibility to unify the trace events is to schedule single path segments in waves. In each step only a single segment is traced, be it for the random walk or the shadow tests. Unfortunately, this approach requires large amounts of intermediate memory (often more than 1 GB) to store all of the hit-points and other information for the paths. Coincidentally, GPUs often have less memory available than there is RAM on CPU side. Also, dispatching threads multiple times imposes an overhead, which usually costs more than the gain in this method.

Therefore, a GPU tailored method should focus on two design goals:

- Short paths of equal length to reduce divergence
- Low additional memory consumption

The idea behind PCLT is to reduce the memory footprint of SPPM while reducing the bias at the same time. Instead of storing millions of photons, only the first view path vertex after a specular or glossy reflection is stored. Storing view path vertices instead of photons itself does not change the outcome of a photon mapper and has been done before [Hachisuka et al. 2008; Havran et al. 2005]. The same sub-paths are merged in any case, because the distance queries are symmetric. A consequence of the reversed radiance estimate is that each contribution of a photon must be added atomically to the output. Usually, all photons in a neighborhood are summed from the same thread, which is the only one operating on the current pixel. In the reversed form, a photon contributes to a random pixel whenever a merge is found. However, this proved to be a small problem in practice and allows to reduce the required memory.

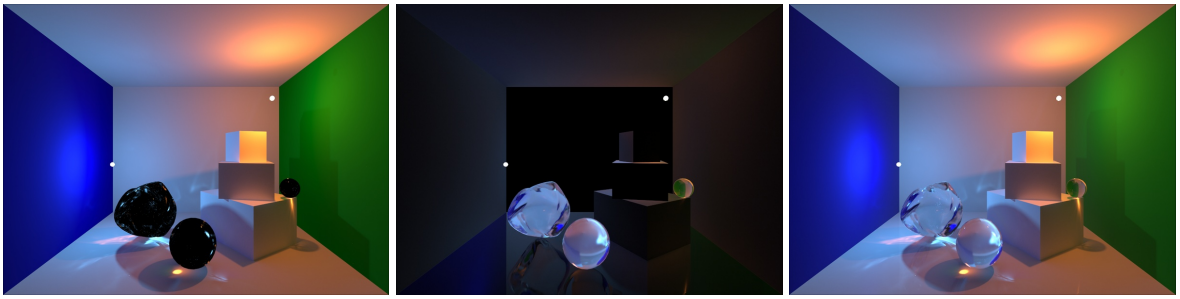
The contributions in PCLT [Jendersie et al. 2017] are:

- A hand-made combination of samplers to avoid storage and computations for MIS
- A fast hash grid implementation for fixed sized neighborhood queries

Thereby, sacrificing the MIS completely is a choice I would not repeat today. An alternative approach is discussed at the end of this chapter.

In retro-perspective the publication was not entirely correct. Back then, the MIS weight computation used for BPT did not use the numerically more robust quotient form. Instead it clamped too large PDF values $p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1})$ to avoid an overflow of the product PDF. This reduced the effectiveness of the MIS and the comparison between PCLT and BPT was not entirely fair. Still, most of the observations hold.

Robust BPT weight Eq. (II.88) p. 76



(a) Light Tracing (b) Pixel Cache (c) Combination (a)+(b)

Figure VII.2: Contributions in PCLT.

VII.1 The Algorithm

The first observation is that Light Tracing is very good at rendering caustics and also produces direct and indirect lighting on diffuse surfaces. It scales well with moderately many lights, because photons are distributed proportional to the irradiance. Since flux is usually shared equally among the photons, the particle density is proportional to the irradiance on the surfaces. For many lights this means that for each light source the most emitted photons are deposited in the dominant region of the respective source. LT Sec. II.8.4 p. 74

Contrarily, a naïve random *Next Event Estimation* would often choose a distant light source with a low contribution. There are better methods for the NEE of many lights, as will be detailed in the discussion. NEE Sec.II.7.4 p. 67

Figure VII.2 (a) shows the result of pure LT in an indoor scenario. The only missing paths are those which have a highly glossy connection towards the camera. Most of these paths can be found easily by a path tracer except the SDS paths for which we need merges. Hence, we trace paths from the observer until they hit a *diffuse* surface and store this hit point for later merges. This storage is called the *Pixel Cache* and is implemented as a hash grid. Since direct illumination is included in LT I decided to not store directly visible *diffuse* hits. The notion of *diffuse* will be detailed in the next subsection.

Combining the paths, the full algorithm consists of two passes:

1. View path tracing. The first diffuse hit after at least one glossy indirection is stored to the *Pixel Cache*.
2. Light path tracing. At any light vertex a connection to the camera and a merge with the *Pixel Cache* is performed. Thereby, only the *diffuse* part of the BSDF is applied in both events.

The contribution from the *Pixel Cache* merges and the full combination of the two contributions are shown in Figure VII.2 (b) and (c). A schematic overview of the used paths is given in Figure VII.3.

VII.1.1 Diffuse Event Decisions

A problem in modern scenes is that there is no clear distinction between diffuse and glossy. For most materials several layers are combined such that they mix diffuse, glossy and specular parts.

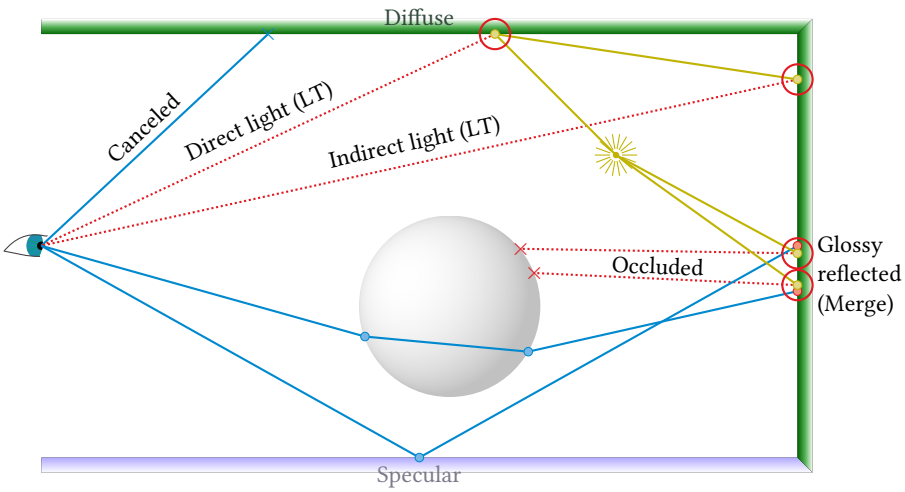
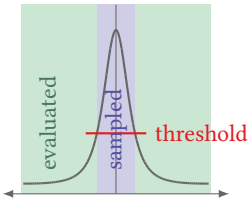


Figure VII.3: Paths used by PCLT. Only glossy reflected view path vertices ● are stored (pixel cache). Other view path vertices ● and light path vertices ● are transitory. A merge ○ is attempted around each light path vertex ● immediately.

In the original implementation I utilized the Russian roulette decision between these layers. If the BSDF sampling returned a diffuse layer the vertex is stored, otherwise the tracing is proceeded. The decision if a layer is diffuse was only based on the used model where Lambertian and Oren-Nayar would count as diffuse. This gives poor results if for example a very rough Microfacet BRDF is employed.

Lambert Sec. II.5.3 p. 44
Oren-Nayar Sec. II.5.4 p. 44
Microfacet Sec. II.5.5 p. 49

Today, I would make use of the current PDF by sampling the BSDF. One option is to split the PDF into two parts. If the value is below a selected threshold, the vertex gets stored. On evaluation in connections and merges one has to check if the PDF for the given directions is above this threshold in which case the result is discarded. This produces a slicing of the BSDF. Large peaks (areas above the threshold) are sampled by recursive tracing while flatter parts are always evaluated. This approach is more flexible and also includes the rough microfacet models into *diffuse* surfaces. It shares the property that we do not need to store values for MIS weights, since for each path there is one and only one position at which it will be terminated. However, it has the drawback that there might be visible discontinuities for different viewing angles on a surface.



Another option is to compute a probability $P_{\text{Continue}} = \left[\frac{(p-\cos\theta/\pi)^+}{p}\right]^\beta$ where p is the PDF of the BSDF sampling. The computation of P_{Continue} is inspired by the power heuristic with respect to an offset Lambertian BRDF. For the Lambertian model P_{Continue} is always zero, so the walk would terminate. For glossy models there is some nonzero probability to proceed the path, which can be used in a Russian roulette decision and weighting process. This approach also generalizes to arbitrary models, generates a single stored vertex (although at varying locations) and does not require an MIS weight. Indeed, the Russian roulette with P_{Continue} already mimics a pre-made MIS weight.

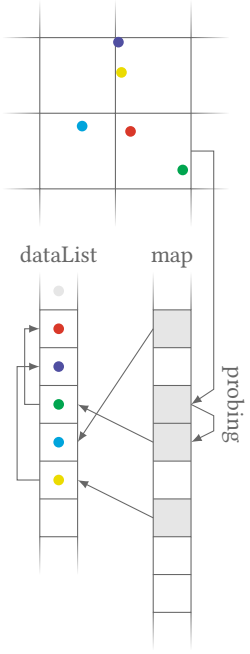
VII.2 A Hash Grid for GPU Neighborhood Queries

The hash grid introduced in the PCLT paper is the data structure I used for every fixed-radius query of photons in this thesis. In fact, it is the predecessor of the density hash grid already explained in Section III.1. It builds on quadratic probing for collisions between different cells and uses a single linked list for the data. This combination was also used in Alcantara's thesis [Alcantara 2011] with good performance.

The hash grid for particle queries consists of two arrays. The first is a densely packed array with the data. It is maintained as an atomic append list and grows by one element for each insertion. The second is the actual map, which stores hash values for identification and the index of the last inserted data element. If multiple particles lie in one cell, they are linked by further indices stored along with the data.

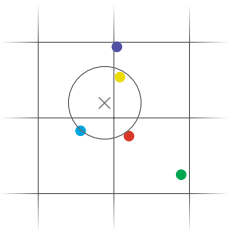
```
type Entry
  Payload data
  u32 next      # Single linked list of data, ~0 marks the end
end

type HashGrid
  Entry[] dataList
  (u32 hash, u32 dataIdx)[] map
  u32 dataCount
  f32[3] cellSize # Spatial granularity of cells
und

func clear(HashGrid grid)
  grid.dataCount = 0  # Forgetting data is easy
  for x in grid.map:  # Map must be overwritten completely
    x = (~0, ~0)      # Empty and end of list markers
  end
end

func insert(HashGrid grid, f32[3] pos, Payload data)
  # Store data in append list
  dataIdx = atomic_add(grid.dataCount, 1)
  grid.dataList[dataIdx].data = Payload
  grid.dataList[dataIdx].next = dataIdx # for the swap later

  # Simple hash similar to a linear congruential generator
  u32[3] cell = <u32>(floor(pos / grid.cellSize))
  u32 hash = dot(cell, [0xb286aff7, 0x35e4a487, 0x75a9c18f])
  s = 0 # Use quadratic probing (begin with step 0)
  while true:
    idx = (hash + (s&1 ? s*s : -s*s) + len(grid.map)) % len(grid.map)
    expected = ~0
    if atomic_cmp_swap(inout grid.map[idx].hash, inout expected, hash)
    or expected == hash:
      # Cell was marked empty or has data with the same hash
      # -> insert to linked list atomically
      atomic_swap(inout grid.map[idx].dataIdx,
                  inout grid.dataList[dataIdx].next)
      break
    end
    # Otherwise proceed searching for an empty or the correct cell
    s += 1
  end
end
```

Hash grid type and insertion



One trick of the implementation is that only collisions between unequal hash values are resolved. The variable `hash` is likely different for various cells. However, it may happen that two cells with different coordinates end up with the same hash in which case all particles of the two cells are stored into one list. This approach of using a single `u32` needs less memory and works better with hardware atomic support than storing full cell coordinates as in the density hash grid. Since the number of such collisions is very low in practice, the performance penalty is rather small.

`DensityHashGrid` III.1 p. 84



To make a query for near particles, all cells which are overlapped by the query region must be iterated. For each cell all linked data entries are enumerated and a distance check is executed. Therefore, the performance of queries depends on the cell size. If cells are much larger than the query radius, a distance check for too many particles is performed. If cells are too small, many hash values must be computed and the number of cache misses increases. A size of two times the query radius is optimal. Then, at most eight cells (in 3D) are overlapped by the query region and the number of particles is still not too large.

Computing the eight hashes of the closest cells is cheap due to the chosen hash function. If the hash of one cell is given, its neighbor cells can be reached by a simple addition of the magic number.

```
u32[3] cell000 = <u32>(round(pos / grid.cellSize))
u32 hash000 = dot(cell000, [0xb286aff7, 0x35e4a487, 0x75a9c18f])
u32 hash001 = hash000 + 0x75a9c18f
...
u32 hash111 = hash000 + 0xb286aff7 + 0x35e4a487 + 0x75a9c18f
```

VII.2.1 Comparison to Stochastic Hash Maps

With the same goal of fast photon queries on a GPU, Hachisuka and Jensen [2010] proposed a *stochastic hash map*. Basically, it is also a hash grid and uses spatial hashing. The different idea is to keep a single random photon per cell instead of resolving collisions. To compensate for the lost energy, it is necessary to count the number of collisions in each map entry. The surviving photon's flux must then be scaled with the number of collisions, because the probability to survive is one divided by the collision count.

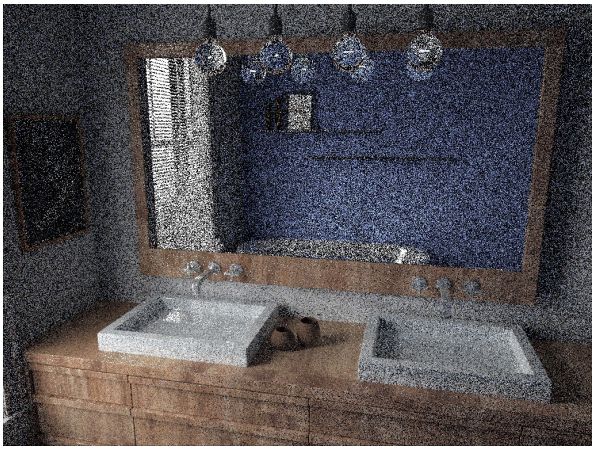
If writing photons, care must be taken to avoid bias. The stochastic map proposed from Hachisuka and Jensen is biased towards longer paths as observed by Davidovič et al. [2014, Sec. 6.2]. A photon may only be written with a probability of $1/(N+1)$ where N is the number of previously stored photons in the cell. This makes sure that each photon has the same chance to survive. A further detail is that a photon is usually larger than a machine word and can therefore not be stored atomically. Instead it is possible to record the photon in memory first and to atomically write its index into the hash map as proposed by Davidovič et al.. Since this would waste the memory, a spin-lock is the better option. A cell can be marked as "in progress" by storing a special index until the entire photon data is stored.

In Figure VII.4 the (unbiased) stochastic hash map is compared with the full map from the previous section. The first thing to be noticed is that photons do not spread over multiple pixels in stochastic hash mapping. This is because in the mirror neighboring pixels have path endpoints which fall

(a) Full hash map 33 MiB

(b) Stochastic hash map 33 MiB

(c) Stochastic hash map 10 MiB



(d) Stochastic hash map 1 MiB

	(a)	(b)	(c)	(d)	+ Tracing
LT + Storing vertices	0.9 ms	0.9 ms	1.0 ms	0.6 ms	+77.4 ms
PT + Query + Estimate	227 ms	97 ms	97 ms	74 ms	+1882 ms

Figure VII.4: Comparison between the full hash map from this section (using a linked list for the data) and stochastic hash maps [Hachisuka and Jensen 2010] of different sizes. Timings are recorded over 8 spp.

into the same hash grid cell. Therefore, at most one pixel in each cell survives the stochastic collision handling. This leads to a visibly higher variance. Note that things are slightly different when storing photons. The inevitable hash collision for photons in the same cell forces to discard more photons in high density regions, which would distribute variance more evenly between bright and dark areas.

Figure VII.4 also reports the timings for hash map related operations. The costs for merges are dominated by the BSDF evaluations for all found photons. Since the full hash map processes more successful merges, the costs increase. In all cases the tracing costs are much larger than the evaluation costs. The hash map operations themselves have negligible impact ($\approx 1\,\mathrm{ms}$) and are practically equal for both hash map types.

Concluding, the full hash map reduces the variance compared to the stochastic map at the cost of more merge evaluations. Thus, in the given application case the full hash map should be preferred. For storing photons the stochastic map may be better, since its memory consumption can be reduced further and the implicit control of photon density might be desirable. The latter depends on the combination of sampling methods. If photons are used for caustics only, the discarding in high density regions would again be a problem and the full map should be used.

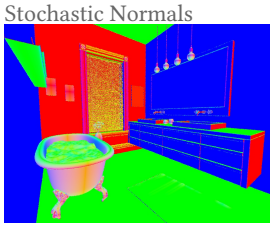
VII.3 Fast Occlusion Tests

In light tracing, each connection to the camera needs to perform an occlusion test – typically in form of an *any-hit* ray test. Since PCLT heavily relies on these contributions, the performance impact of these queries is relatively high. In the Bathroom scenario 25% of the entire iteration time fall upon this connection test.

To reduce occlusion query times we can use a trick from rasterization. It is simple to generate a depth buffer from the first hit points of the view paths. Due to sub-pixel jittering the value will be random within all possible depths in this pixel, so I will name this buffer a *stochastic depth buffer*. Then, a vertex which should be projected can be tested against this random depth value like in shadow mapping. This replaces an entire ray cast with a single memory fetch.

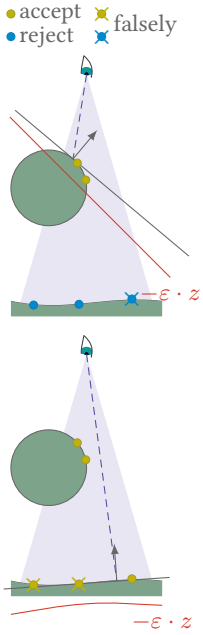
Stochastic Depth

Stochastic Normals



However, the result of such a simple depth test is biased for two reasons. First, even on a fully visible plane vertices are discarded, if a smaller depth value is stored. A similar artifact is known as *shadow acne* (jagged shadows on planes) in shadow mapping algorithms. Second, the stochastic nature may allow occluded values to be falsely accepted. Assume an edge inside a pixel: if the foreground-depth is stored, all background vertices are discarded. Otherwise, all vertices are accepted, including those which are truly occluded. This partially compensates the lost energy from passes with full occlusion.

The error made by the first artifact is much worse than the stochastic occlusion bias at edges. We can solve this problem by storing normals together with the depth values (for the same first hit points). To perform the occlusion test the distance to the plane is computed. The projection is accepted if this distance is greater than $-\varepsilon \cdot z$ where z is the view distance. The small negative value fixes issues on convex bended surfaces. Thereby, ε should be chosen proportional to the tangent of the camera's field of view divided by the resolution. z times ε is then the pixel radius in the query distance.



The plane based occlusion test for projections was also used before for *Eye Path Reprojections* [Henrich et al. 2011]. The authors used a high resolution position and normal buffer to reduce issues with anti-aliasing. Probably, the stochastic version here is less erroneous, but both versions are biased in a similar way.

Figure VII.5 compares bias and performance in the bathroom scene. The plane-based test is much less biased than a pure z test while having the same speed. The difference to the ground truth is negligible. Therefore, the 25% faster plane-based occlusion test should be preferred over the ray test.

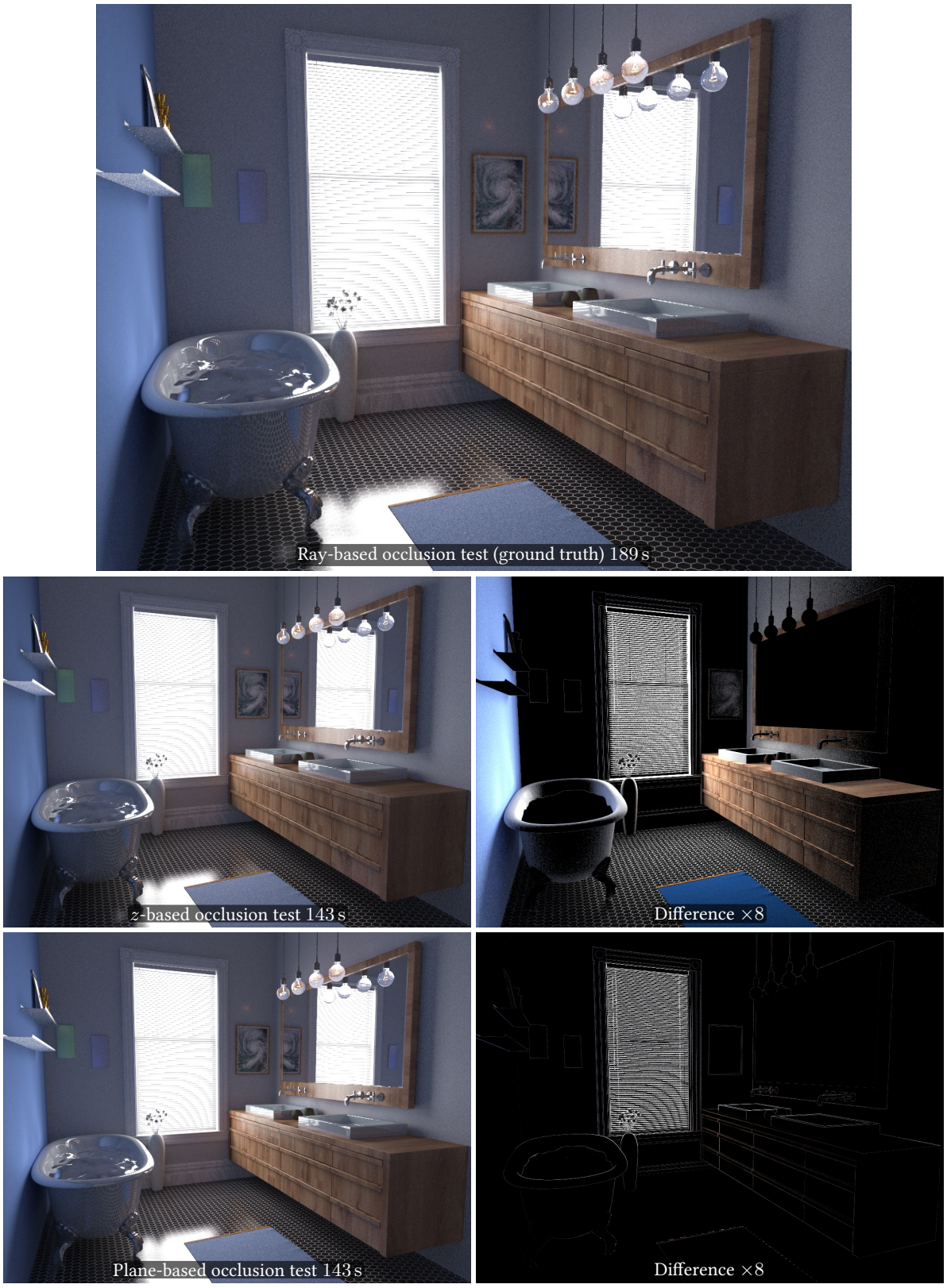


Figure VII.5: Bias and performance of z-based and plane-based occlusion tests. All images are rendered with 5000 spp. The timing improves significantly opposed to ray queries, while the error in the plane-based test is acceptable.

VII.4 Evaluation

The decision to use light tracing and merging only has severe consequences for the strength and weaknesses of this method:

✓ High troughput on GPUs

✓ Includes caustics and SDS paths

✓ Scales well with many lights

X Bad for large distance connections

X Unnecessary high noise in direct illumination

Before discussing these (dis)advantages a first interesting question is, whether it is worth to distribute more light paths than view paths under the selected choices for PCLT. Since the memory consumption depends on the number of glossy and specular pixels, the number of photons can be increased at no additional memory costs.

Figure VII.6 compares SPPM with PCLT with varying numbers of photons per view path. The number of traced photons is the same for all images. Both PCLT and SPPM are significantly faster when using more photons per iteration. With the enabled plane-based occlusion test, PCLT is only slightly slower than SPPM. Even though there are less view paths, the quality remains very similar. This, however, might not be true for other scenarios with glossy surfaces, where view paths produce the majority of contributions.

With increasing photon count, the memory consumption is increasing for SPPM while it remains constant for PCLT. Contrarily, the memory for the pixel cache grows with image resolution. At FullHD resolution 82 MiB would be required instead of the 33 MiB. However, the sizes of the pixel cache are chosen conservatively because the number of glossy pixels is not known before. Allocating a smaller map would work well for most cases. In cases where there would be an overflow, the mapping could fall back to stochastic hash mapping, ensuring energy preservation.

VII.4.1 Noise

PCLT uses only a few selected samplers and thus has a relatively high noise level. Referring back to Figure VII.1 we can see that the direct illumination is handled better by most other methods. PT and BPT mostly do direct illumination over the next event estimation. In SPPM, the blurring of photons reduces the variance. This is also visible in Figure VII.6. Here, SPPM has the smaller variance, but the noise is also more structured. In direct comparison with SPPM, the noise in PCLT is to be preferred. It not only avoids the bias in direct illumination, but it is also better suited for denoising[1].

The situation for PCLT becomes worse for increased transport distances. As shown in Figure VII.7, a simple path tracer can outperform PCLT by far. For this reason it would make sense to reintroduce NEE into PCLT.

[1] Most denoising algorithms are developed to remove per-pixel noise. Lower frequency signals are assumed to be wanted image signals.

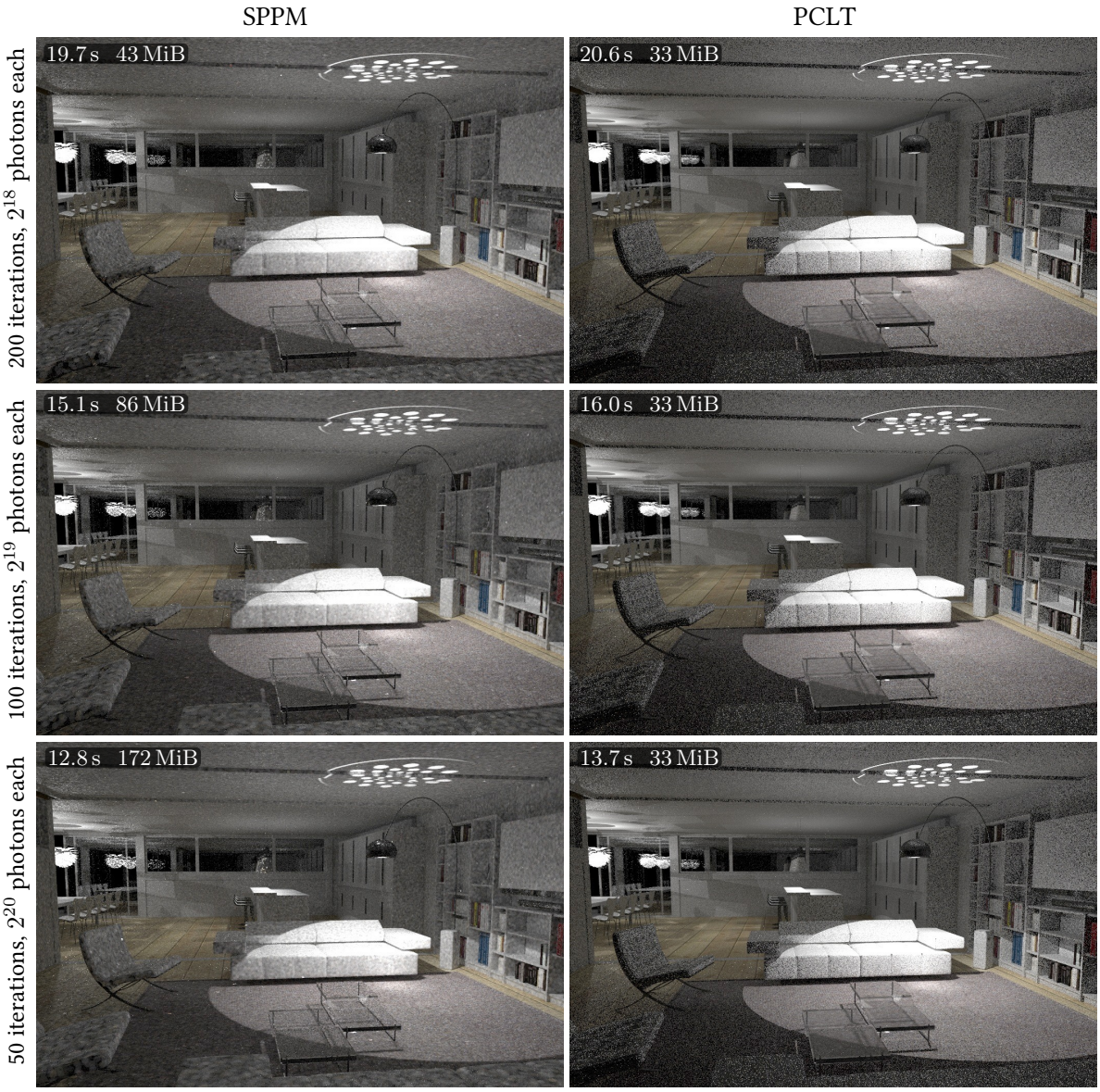
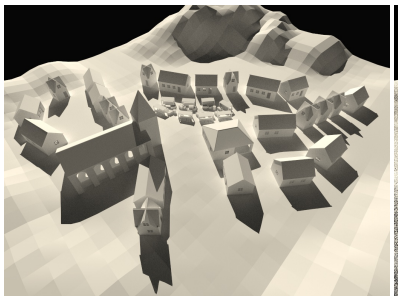
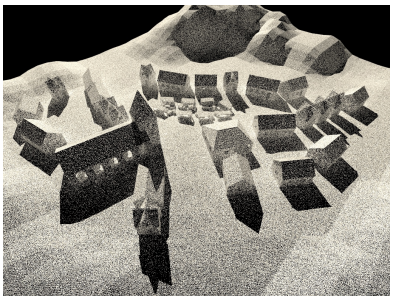


Figure VII.6: Equal light path count comparison. Using more photons per iteration improves performance for both methods, while having almost the same quality. Memory requirements increase for SPPM only.



PT, 1000 spp, 47.5 s

Figure VII.7: Failure case for PCLT: large connection distances.



PCLT, 1000 iterations, 10.9 s

PCLT, 1000 iterations, 258 s

Figure VII.8: Many lights in PCLT.

VII.4.2 Many Lights

As mentioned before, LT is good for moderately many lights, because it distributes the photons proportionally to the irradiance. Regions closer to one light source will receive more photons from this source than distant regions. Figure VII.8 shows an example with 69 light sources. However, for too many lights, LT will become bad again. If the number of light sources is in the number of photon paths per iteration, there will be only one photon per source. This means that LT becomes ineffective again, if the number of light sources is roughly above 10k for typical setups.

VII.5 Discussion: A Better Path Combination

The initial decision to remove NEE from the available samplers has two consequences: One the one hand it increases the throughput of iterations, on the other hand it causes failures in many common scenarios. It thus seems to be a very good idea to reintroduce the NEE to PCLT again. Direct visible lighting would then be a combination of LT and NEE and the contributions from the pixel cache would combine NEE and merges.

Fortunately, the memory consumption remains equal under this change, even though we need to compute MIS weights. In case of direct illumination, the vertices are not stored anyway. The MIS weights for direct illumination can be computed from the known values in the moment of connection. For the pixel cache, there are now two possible samplers: the merge and the NEE. The difference between these two paths is the acceptance probability p_{acc} which requires the path density for the last segment of the light pass. In both cases, this quantity is known without needing to store it with the vertices.

p_{acc} Eq. (II.92) p. 79

Adding back the NEE can also improve the handling of a massive amount of light sources. It is possible to select light sources proportional to the contributions by using search trees [Dachsbacher et al. 2014; Moreau and Clarberg 2019; Novák 2014]. Hence, NEE will connect more often to close or bright light sources. This contribution-based NEE scales even further than the light tracer.

Chapter VIII
Conclusions

In this thesis I proposed several independent improvements to increase the robustness in modern Monte Carlo-based renderers. Most importantly, the variance reduction in MIS weights due to more correct reuse factors and the microfacet regularization in difficult paths.

Reuse Factors Chpt. IV p. 101

Especially the footprint-based estimation of the sample count factor under reuse [Jendersie 2019b] is a solution I would recommend for practical applications. It reliably reduces the variance in photon mapping-based algorithms at a very small computational overhead.

Footprint-based factors Sec. IV.3.1 p. 108

The microfacet regularization approach is a simple and promising modification for rendering algorithms. Without guidance or similar additional improvements it is not sufficient to produce caustics in a Path Tracer. However, if guidance is included and the control variate approach discussed in Section V.2.4 is used, I expect that regularization can successfully replace photon transport in lightweight renderers. Even without further modification, regularization is applicable in production renderers, which are based on PT, to partially recover the otherwise missing caustic effects.

Microfacet regularization Chpt. V p. 129

The other two approaches, Next Event Backtracking and Pixel Cache Light Tracing, present partial solutions to resolve some of the existing problems. In NEB I proposed an entirely new operation to create light paths. It is extremely efficient for situations in which the light source has a medium or large distance compared to the remaining path length. Moreover, it offers several interesting modifications which might lead to a good transport operator for different difficult scene operations. In the current form it supplements the usual photon transport from light sources well.

NEB Chpt. VI p. 157

The PCLT algorithm focuses on the reduction of computational overhead and memory consumption with the target of GPU architectures. While some of the assumptions I made were due to missing experience on my side, other observations still hold. For one, it is possible to store either photons or importons (view path vertices) in any merge-based sampler. Storing importons can reduce the memory footprint but leads to a scattered write which must be protected with atomics. The other observation is that we can get good results with a hand-selected number of samplers. Opposed to BPT or VCM for which the number of samplers grows with the path length, PCLT only uses a constant number. The usage of a small number of samplers decreases the overhead and scales better on parallel hardware, although I would use a different selection of samplers today.

PCLT Chpt. VII p. 173

The proposed methods are connected by two common concepts: MIS weights and footprints. Any change to a sampler requires a change of the MIS weights. This includes Russian roulette, shading normal correction and the proposed modifications like regularization. The easiest way to find the necessary adaptations is to review the changes of the contribution value, i.e. the primary quantity which we integrate. Actually, using the contribution is formally correct and the usage of path probability densities is only equivalent as long as the integrand does not change.

Russian roulette Sec. II.7.3 p. 65
Shading normal correction Sec. II.4.1 p. 36
Contribution MIS Sec. II.7.6 p. 69

The footprint I deem one of the most interesting estimates for future research. In my thesis I applied them for the improved MIS weights and for one of the heuristics in regularization to hide the bias. Beyond that they can be used for adaptive photon mapping and anti-aliasing. The footprints I derived here are designed to have a lower overhead than the 5D Covariance tracing [Belcour et al. 2013]. However, they lack precision in some parts and are not applicable to all problems.

Footprints Sec. IV.3.2 p. 109

VIII.1 Future Work

From all I have learned, the following lists important, open problems for rendering to be solved. For detailed improvements of the methods in this thesis I refer to the respective discussion sections.

Optimal MIS weights A good MIS weight maximizes the gain of the expensive sampling process. It is a known fact that neither the balance heuristic nor the power heuristic give optimal results. While the recently found optimal weights [Kondapaneni et al. 2019] solve the problem theoretically, the practical application is approximate and scales bad for many techniques. Thus, other new estimates remain an interesting topic.

Indeed, I would apply observations similar to that used for the reuse factor. By applying footprint estimates to guess the variance, it might be possible to compute weights close to the optimal one directly.

Bias in MIS weights Another open problem in the computation of MIS weights is to minimize bias and variance at the same time. The current derivations all focus on a variance minimization alone. A weight which includes both would especially useful for all kinds of path regularizations.

Exploration of regularization Regularization is a promising concept for the use in lightweight renderers and to limit the maximum variance of a path. While the roughness-based approach works for most material models in use, a black box solution for arbitrary materials is desirable. The most promising candidate is the discussed control variate approach using closed form approximate solutions, or even the simpler direct use of the approximation.

Besides materials, the length of path segments contributes to the final variance. In connections, a possible regularization is to offset the distance by which we divide. This would cause a systematic loss of energy, but might be useful for truly limiting the maximum variance.

Heuristics for the amount of regularization Independent of the operations to regularize the path, adaptive scaling should be applied. The

goal is to minimize the total error of a path (bias + variance) to select an appropriate amount of regularization. This is similar to the introduction of bias into MIS weights, because both need estimates of the samplers variance and bias.

When designing heuristics, special care must be taken to guarantee temporal coherence. This property is required by production renderers to avoid flickering in animated scenes. The sampler quality heuristic from Section V.5.1 does not fulfill this requirement, yet.

Footprints The density of paths of a certain sampler is one of the most important properties to assess bias and variance as required in the above proposals. Thereby, the research for footprints should target two criteria: precision and fast computation. From what I learned, it seems to be necessary to use anisotropic estimates for an improved precision, while simple scalar estimates are faster to calculate. The main difficulty is still the unification of microscopic scattering from the BSDF and macroscopic scattering from changes in the scene.

Chapter A
Appendix

A.1 Jacobians for the Transformation of PDFs

Any importance sampling function used in this thesis can be seen as a transformation of a set of uniform distributed variables into some other domain. In section II.2.6 this domain transformation is described. This section shows the derivation of the Jacobians used in the various sampling routines.

Domain change II.2.6 p. 28

A.1.1 Scaling Operator

If a vector $\boldsymbol{u}$ is scaled by a component wise product with a scaling scalar or vector $\boldsymbol{a}$ the inverse function and its Jacobian $|\det \boldsymbol{J}|$ are straight forward:

$$\boldsymbol{u} = \boldsymbol{x} \oslash \boldsymbol{a}$$

$$\left\| \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \right\| = \begin{vmatrix} \frac{1}{a_0} & 0 & \cdots \\ 0 & \frac{1}{a_1} & \cdots \\ \vdots & \vdots & \ddots \end{vmatrix} = \prod_{i=0}^{k-1} \frac{1}{a_i}$$

$\square \oslash \square$: component wise (Hadamard) division

where i iterates over the vector components in k dimensions. In case of a scalar scaling factor the result is simply $1/a^k$.

A.1.2 Polar to Cartesian Coordinate

For a vector given in polar coordinates we compute the vector in Cartesian coordinates as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \sin\theta \cos\phi \\ r \sin\theta \sin\phi \\ r \cos\theta \end{bmatrix}$$

by the usual convention. The inverse of this transformation is

$$\begin{bmatrix} \theta \\ \phi \\ r \end{bmatrix} = \begin{bmatrix} \arccos(z/\sqrt{x^2+y^2+z^2}) \\ \arctan(y, x) \\ \sqrt{x^2+y^2+z^2} \end{bmatrix}$$

where we can use $\arctan(y, x) = \arctan(y/x)$ for the purpose of derivations (to compute the inverse the two argument tangent must be used). The determinant of the Jacobian is then

$$\left\| \frac{\partial(\theta, \phi, r)}{\partial(x, y, z)} \right\| = \left\| \begin{matrix} \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial y} & \frac{\partial\theta}{\partial z} \\ \frac{\partial\phi}{\partial x} & \frac{\partial\phi}{\partial y} & \frac{\partial\phi}{\partial z} \\ \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \end{matrix} \right\| = \left| \frac{1}{\sqrt{x^2+y^2+z^2}\sqrt{x^2+y^2}} \right| = \left| \frac{1}{r^2 \sin\theta} \right| \tag{A.1}$$

The last equality is obtained by inserting the forward transformation from polar to Cartesian coordinates again. For normalized vectors the radius $r = \sqrt{x^2+y^2+z^2} = 1$ vanishes.

A.1.3 NORMALIZATION OPERATOR

The normalization operator from slope space to a direction vector

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = \frac{(u, v, d)^T}{\sqrt{u^2+v^2+d^2}}$$

is used in many direction samplers, like for *Normals Distribution Functions* (sNDFs) and the camera models. It is important to have one fixed component (here the d) to be able to find the inverse transformation back to the parameter space. For the slope space and the pinhole camera we have $d = 1$, whereas the thin lens camera uses an arbitrary, but still constant d.

Pinhole camera Sec. II.6.1 p. 59
Thin lens camera Sec. II.6.2 p. 60

Effectively, the inverse is the vector $(x, y, z)^T$ multiplied with the length $\sqrt{u^2+v^2+d^2}$ which we get from the knowledge that $z = d/\sqrt{u^2+v^2+d^2}$:

$$u = d\frac{x}{z} = d\frac{\sin\theta\cos\phi}{\cos\theta}$$
$$v = d\frac{y}{z} = d\frac{\sin\theta\sin\phi}{\cos\theta}$$

Unfortunately, x, y and z depend on each other because of the normalization condition $1 = x^2+y^2+z^2$ which complicates the derivation of the Jacobian in Cartesian coordinates. Instead, we go to spherical coordinates first and then use the Polar to Cartesian transformation from the previous section (Equation (A.1) with $r = 1$ by construction):

$$\begin{aligned} \left\| \frac{\partial(u, v)}{\partial(x, y)} \right\| &= \left\| \frac{\partial(u, v)}{\partial(\theta, \phi)} \right\| \cdot \left\| \frac{\partial(\theta, \phi, r)}{\partial(x, y, z)} \right\| \\ &= \left\| \begin{matrix} \frac{\partial u}{\partial\theta} & \frac{\partial u}{\partial\phi} \\ \frac{\partial v}{\partial\theta} & \frac{\partial v}{\partial\phi} \end{matrix} \right\| \cdot \left| \frac{1}{\sin\theta} \right| \\ &= \left| \frac{d^2 \sin\theta}{\cos^3\theta} \right| \cdot \left| \frac{1}{\sin\theta} \right| \\ &= \frac{d^2}{|\cos\theta|^3} \end{aligned} \tag{A.2}$$

Following the convention from the previous section $\cos\theta$ is equal to the z component of the normalized vector. If a routine evaluates the PDF for a

global space direction, it is computed by the dot product between the normal of a surface and the respective direction.

A similar derivation can be found in the supplemental of Dupuy et al. [2013].

A.2 Tables

A.2.1 Oren-Nayar Albedo

In Section II.5.8 I proposed to simply rescale the Oren-Nayar model for artistic reasons. For the required average albedo ϱ the numerical values from the following table can be used.

Roughness α	Albedo ϱ
0	1
$\pi/32$	0.9984515433
$2\pi/32$	0.9868863928
$3\pi/32$	0.9600609238
$4\pi/32$	0.9232777349
$5\pi/32$	0.8841291491
$6\pi/32$	0.8474927155
$7\pi/32$	0.8154250323
$8\pi/32$	0.7883002593
$9\pi/32$	0.7657241714
$10\pi/32$	0.7470450606
$11\pi/32$	0.7315916234
$12\pi/32$	0.7187656605
$13\pi/32$	0.7080666669
$14\pi/32$	0.6990884661
$15\pi/32$	0.6915061377
$16\pi/32$	0.6850612128
$17\pi/32$	0.6795481399
$18\pi/32$	0.6748032291
$19\pi/32$	0.6706953859
$20\pi/32$	0.6671192049
$21\pi/32$	0.6639894329
$22\pi/32$	0.6612367028
$23\pi/32$	0.658804242
$24\pi/32$	0.6566453051
$25\pi/32$	0.6547211818
$26\pi/32$	0.6529996354
$27\pi/32$	0.6514536737
$28\pi/32$	0.6500605773
$29\pi/32$	0.648802979
$30\pi/32$	0.6476608432
$31\pi/32$	0.6466202214
π	0.6456728566

A.3 Proofs

A.3.1 Bounded Path Variance

If the relative variance of each event on a path is bounded by τ: Used in Eq. (V.2) p. 131

$$\begin{aligned} &\frac{\mathrm{V}[X]}{\mathrm{E}[X]^2} \leq \tau \\ \Leftrightarrow\ &\mathrm{V}[X] \leq \tau\,\mathrm{E}[X]^2\,, \end{aligned} \tag{A.3}$$

then the relative path variance is bounded by

$$\tau_\ell = \sum_{i=1}^{\ell+1} \binom{\ell+1}{i} t^i$$

which can be proven by induction.

The smallest valid path has $\ell = 1$ and is that of a directly visible light source. It has two vertices – one at the camera and another on the light source – which we associate with the events W and L. Either of the two can be sampled or evaluated and we do not care which terms of the sampler get assigned to which event. All we need is that both are bounded ($\mathrm{V}[W]/\mathrm{E}[W]^2 \leq \tau$ and $\mathrm{V}[L]/\mathrm{E}[L]^2 \leq \tau$). The variance of the full path is

$$\begin{aligned} \mathrm{V}[WL] &= \mathrm{V}[W]\,\mathrm{E}[L]^2 + \mathrm{E}[W]^2\,\mathrm{V}[L] + \mathrm{V}[W]\,\mathrm{V}[L] && \text{Eq. (II.16c) p. 20 applied} \\ &\leq \tau\,\mathrm{E}[W]^2\,\mathrm{E}[L]^2 + \tau\,\mathrm{E}[W]^2\,\mathrm{E}[L]^2 + \tau^2\,\mathrm{E}[W]^2\,\mathrm{E}[L]^2 && \text{Insertion of Eq. (A.3)} \\ &= (2\tau + \tau^2)(\mathrm{E}[W]\,\mathrm{E}[L])^2 && \text{Distributivity} \\ &= (2\tau + \tau^2)\,\mathrm{E}[WL]^2 && \text{Eq. (II.12c) p. 19 applied} \end{aligned}$$

Hence the relative variance is

$$\frac{\mathrm{V}[WL]}{\mathrm{E}[WL]^2} \leq 2\tau + \tau^2 = \sum_{i=1}^{2} \binom{2}{i} \tau^i = \sum_{i=1}^{\ell+1} \binom{\ell+1}{i} \tau^i$$

which proves the base case. In the induction step we can apply the very same steps as above by extending a path $\mathcal{P}^\ell$ by an additional random walk segment X. In the measurement contribution function, X is the value $\rho \cdot G$ which is determined by the single random choice of a direction.

$$\begin{aligned} \mathrm{V}\left[\mathcal{P}^{\ell+1}\right] = \mathrm{V}\left[\mathcal{P}^\ell X\right] &= \mathrm{V}\left[\mathcal{P}^\ell\right]\mathrm{E}[X]^2 + \mathrm{E}\left[\mathcal{P}^\ell\right]^2\mathrm{V}[X] + \mathrm{V}\left[\mathcal{P}^\ell\right]\mathrm{V}[X] \\ &\leq \tau_\ell\,\mathrm{E}\left[\mathcal{P}^\ell\right]^2\mathrm{E}[X]^2 + \tau\,\mathrm{E}\left[\mathcal{P}^\ell\right]^2\mathrm{E}[X]^2 + \tau\tau_\ell\,\mathrm{E}\left[\mathcal{P}^\ell\right]^2\mathrm{E}[X]^2 \\ &= \mathrm{E}\left[\mathcal{P}^\ell X\right]^2(\tau + \tau_\ell + \tau\tau_\ell) \end{aligned}$$

This yields a relative variance for the extended path of

$$\frac{\mathrm{V}\left[\mathcal{P}^{\ell+1}\right]}{\mathrm{E}\left[\mathcal{P}^{\ell+1}\right]^2} \leq \tau + \tau_\ell + \tau\tau_\ell$$

$$= \tau + \sum_{i=1}^{\ell+1} \binom{\ell+1}{i} \tau^i + \sum_{i=1}^{\ell+1} \binom{\ell+1}{i} \tau^{i+1}$$ Insert τ_ℓ from Eq. (A.3.1)

$$= \left(1 + \binom{\ell+1}{1}\right) \tau + \sum_{i=2}^{\ell+1} \binom{\ell+1}{i} \tau^i + \sum_{i=1}^{\ell} \binom{\ell+1}{i} \tau^{i+1} + \binom{\ell+1}{\ell+1} \tau^{\ell+2}$$ Extract the first term from the first sum and the last term from the second sum.

$$= \binom{\ell+2}{1} \tau + \sum_{i=2}^{\ell+1} \binom{\ell+1}{i} \tau^i + \sum_{i=2}^{\ell+1} \binom{\ell+1}{i-1} \tau^i + \binom{\ell+2}{\ell+2} \tau^{\ell+2}$$ Equiv. transform the τ and $\tau^{\ell+2}$ terms and change the indexing of the 2nd sum.

$$= \binom{\ell+2}{1} \tau + \sum_{i=2}^{\ell+1} \left(\binom{\ell+1}{i-1} + \binom{\ell+1}{i} \right) \tau^i + \binom{\ell+2}{\ell+2} \tau^{\ell+2}$$ Merge the two sums.

$$= \binom{\ell+2}{1} \tau + \sum_{i=2}^{\ell+1} \binom{\ell+2}{i} \tau^i + \binom{\ell+2}{\ell+2} \tau^{\ell+2}$$ Apply a property of binomial coefficient.

$$= \sum_{i=1}^{\ell+2} \binom{\ell+2}{i} \tau^i$$ Include the first and the last term into the sum.

■

Note that, if the absolute variance should be bounded, we can move the expected value of the path to the other side of the equation as done in (A.3). This means if we know the expected value, we can as well bound the absolute variance. However, the expected value is exactly the value we search in the integration process. Therefore, we would need to know the final result before we are able to compute a useful bound for a single event.

A.3.2 VOLUME OF AN AXIS-ALIGNED SIMPLEX

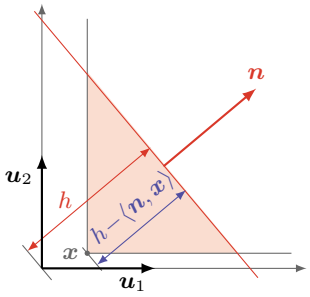
The volume of a general d-dimensional simplex is

$$V(\Delta) = \frac{|\det(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d)|}{d!} \tag{A.4}$$

where $\boldsymbol{e}_k$ are the edges $\boldsymbol{x}_k - \boldsymbol{x}_0$ of the simplex.

In Section III.1.1 p. 85 we are interested in the volume of simplexes for which all $\boldsymbol{e}_i$ are aligned with the coordinate axis:

$$\Delta(h, \boldsymbol{x}) = \left\{ \boldsymbol{y} = (y_1, \ldots, y_d) \in \mathbb{R}^d : \langle \boldsymbol{n}, \boldsymbol{y} \rangle \leq h \wedge \forall k \in [1, d] : y_k \geq x_k \right\}$$



All sides of these simplexes are aligned with the coordinate planes, except the 'outer' plane which is defined by $\boldsymbol{n}$ and h. Thereby, the origin $\boldsymbol{x}_0$ of the simplex $\Delta(h, \boldsymbol{x})$ is $\boldsymbol{x}$.

The other vertices lie on the intersection between the specified plane ($\langle \boldsymbol{n}, \boldsymbol{y} \rangle - h = 0$) and the rays $\boldsymbol{x}_k = \boldsymbol{x} + t_k \cdot \boldsymbol{u}_k$ where $\boldsymbol{u}_k$ are the unit vectors of the coordinate axis. Inserting the ray into the plane equation yields

$$\begin{aligned}
& \langle \boldsymbol{n}, \boldsymbol{x} + t_k \cdot \boldsymbol{u}_k \rangle = h \\
\Leftrightarrow \quad & \langle \boldsymbol{n}, \boldsymbol{x} \rangle + t_k \langle \boldsymbol{n}, \boldsymbol{u}_k \rangle = h \\
\Leftrightarrow \quad & t_k = \frac{h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle}{\langle \boldsymbol{n}, \boldsymbol{u}_k \rangle} \\
\Leftrightarrow \quad & t_k = \frac{h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle}{n_k}
\end{aligned}$$

where t_k is the length of the edge $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x} = t_k \cdot \boldsymbol{u}_k$. Note that we must now introduce the clamping $\max(0, h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle)/n_k = (t_k)^+$ to get a zero volume simplex if the plane is below the support position $\boldsymbol{x}$.

Since all edges in our simplex are aligned with the coordinate axis $\boldsymbol{u}$, the matrix $[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d] = [(t_1)^+ \boldsymbol{u}_1, \ldots, (t_d)^+ \boldsymbol{u}_d]$ is a diagonal matrix. The determinant of this matrix is simply the product of its diagonal entries $(t_k)^+$.

$$\begin{aligned}
V(\Delta(h, \boldsymbol{x})) &= \frac{1}{d!} \prod_{k=1}^{d} (t_k)^+ = \frac{1}{d!} \prod_{k=1}^{d} \frac{\max(0, h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle)}{n_k} \\
&= \frac{1}{d!} \frac{\prod_{k=1}^{d} \max(0, h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle)}{\prod_{k=1}^{d} n_k} \\
&= \frac{1}{d!} \frac{\max(0, h - \langle \boldsymbol{n}, \boldsymbol{x} \rangle)^d}{\prod_{k=1}^{d} n_k}
\end{aligned} \tag{A.5}$$

A.3.3 Ratio of Sampler Variance without and with Reuse

In Section IV.3 the ratio of variances of a sampler without and with reuse is used (Eq. (IV.8) p. 107). For any sampler the terms were grouped into three parts: V (view sub-path sampling), L (light sub-path sampling) and C (remaining terms). Using this notation the variance of a Monte Carlo sampler is

$$\begin{aligned}\mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N} V_i C_i L_i\right] &\equiv \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{V}[V_i C_i L_i] \equiv \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{V}[VCL] \\ &\equiv \frac{1}{N}\mathrm{V}[VCL] \\ &\equiv \frac{1}{N}\left(\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \mathrm{E}[V]^2\,\mathrm{V}[CL] + \mathrm{V}[V]\,\mathrm{V}[CL]\right).\end{aligned}$$

The indices i can be removed since the variance of a function does not depend on the value of a single sample $\mathrm{V}[f_i] = \mathrm{V}[f]$.

Now, we analogously estimate the variance of a sampler with reuse. This means a part of the values is averaged over additional sub-samples:

$$\mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N} V_i \frac{1}{N_\Phi}\sum_{j=1}^{N_\Phi} C_{ij}L_{ij}\right] \equiv \frac{1}{N^2 N_\Phi^2}\sum_{i=1}^{N}\mathrm{V}\left[V_i\sum_{j=1}^{N_\Phi} C_{ij}L_{ij}\right]$$

Moving scalar factors Eq. (II.16b) p. 20

$$\equiv \frac{1}{N^2 N_\Phi^2}\sum_{i=1}^{N}\left(\mathrm{V}[V_i]\,\mathrm{E}\left[\sum_{j=1}^{N_\Phi} C_{ij}L_{ij}\right]^2 + \left(\mathrm{E}[V_i]^2 + \mathrm{V}[V_i]\right)\cdot\mathrm{V}\left[\sum_{j=1}^{N_\Phi} C_{ij}L_{ij}\right]\right)$$

Product variance Eq. (II.16c) p. 20 + Distributivity on the resulting scalars

$$\equiv \frac{1}{N^2 N_\Phi^2}\sum_{i=1}^{N}\left(\mathrm{V}[V_i]\left(\sum_{j=1}^{N_\Phi}\mathrm{E}[C_{ij}L_{ij}]\right)^2 + \left(\mathrm{E}[V_i]^2 + \mathrm{V}[V_i]\right)\left(\sum_{j=1}^{N_\Phi}\mathrm{V}[C_{ij}L_{ij}]\right)\right)$$

Moving sums from expected value Eq. (II.12a) p. 19 and variance Eq. (II.16a) p. 20

$$\equiv \frac{1}{N^2 N_\Phi^2}\sum_{i=1}^{N}\left(\mathrm{V}[V]\left(\sum_{j=1}^{N_\Phi}\mathrm{E}[CL]\right)^2 + \left(\mathrm{E}[V]^2 + \mathrm{V}[V]\right)\left(\sum_{j=1}^{N_\Phi}\mathrm{V}[CL]\right)\right)$$

Removing indices, because of independence of E and V from actual sample values

$$\equiv \frac{1}{N N_\Phi^2}\left(\mathrm{V}[V]\cdot N_\Phi^2\cdot\mathrm{E}[CL]^2 + \left(\mathrm{E}[V]^2 + \mathrm{V}[V]\right)\cdot N_\Phi\cdot\mathrm{V}[CL]\right)$$

Resolving sums of identical values

$$\equiv \frac{1}{N}\left(\mathrm{V}[V]\,\mathrm{E}[CL]^2 + \frac{1}{N_\Phi}\left(\mathrm{E}[V]^2 + \mathrm{V}[V]\right)\mathrm{V}[CL]\right).$$

Canceling N_Φ

Finally, dividing the two results yields the formulation given in Equation (IV.8) where $1/N$ cancels out. ■

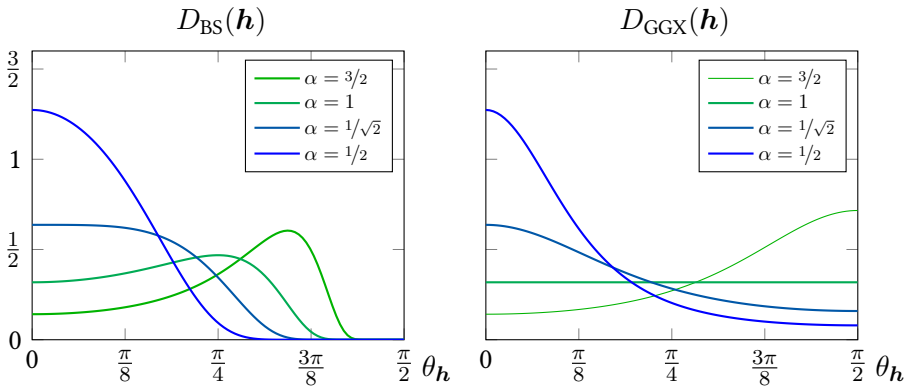


Figure A.1: Plots of NDFs for several roughness values α.

A.3.4 Maximum Values of NDFs

In the following the extrema of the common NDF functions are determined through setting $\partial D/\partial\theta = 0$, where $\theta \in [0, \frac{\pi}{2}]$ is the angle between half vector $\boldsymbol{h}$ and normal $\boldsymbol{n}$.

For the GGX distribution we get

$$\frac{\partial D_{\mathrm{GGX}}}{\partial\theta} = \frac{2\alpha^2(\alpha^2-1)\sin(2\theta)}{\pi((\alpha^2-1)\cos^2(\theta)+1)^3} \tag{A.6}$$

D_{GGX} Eq. (II.55a) p. 50

with roots $\theta = \{0, \pi/2\}$ within the valid domain. For $\alpha \in [0, 1]$ the values at these roots are maximum and minimum respectively:

$$D_{\mathrm{GGX}}(\theta = 0) = \frac{1}{\pi\alpha^2} \qquad D_{\mathrm{GGX}}\left(\theta = \frac{\pi}{2}\right) = \frac{\alpha^2}{\pi} \tag{A.7}$$

For the Beckmann-Spizzichino distribution we get

$$\frac{\partial D_{\mathrm{BS}}}{\partial\theta} = -\frac{2\tan(\theta)\exp(-\tan^2(\theta)/\alpha^2)(\cos^{-2}(\theta)-2\alpha^2)}{\pi\alpha^4\cos^4\theta} \tag{A.8}$$

D_{BS} Eq. (II.56a) p. 51

with roots $\theta = \{0, \arccos(1/\alpha\sqrt{2})\}$. Whether the two points

$$D_{\mathrm{BS}}(\theta = 0) = \frac{1}{\pi\alpha^2} \quad D_{\mathrm{BS}}\left(\cos\theta = \frac{1}{\alpha\sqrt{2}}\right) = \frac{4\alpha^2\exp(\alpha^{-2}-2)}{\pi} \tag{A.9}$$

are a maximum or a minimum depends on alpha: The value at $\theta = 0$ is the maximum for $\alpha \in [0, 1/\sqrt{2}]$ and becomes a local minimum for larger alphas. Then, the second value becomes the local maximum. Also see Figure A.1 as visual reference.

Finally, for the Cosine distribution we get

$$\frac{\partial D_{\mathrm{Cos}}}{\partial\theta} = \frac{(2/\alpha^2-2)\cos(\theta)^{2/\alpha^2-3}\sin(\theta)}{\pi\alpha^2} \tag{A.10}$$

D_{Cos} Eq. (II.57) p. 51

with roots $\theta = \{0, \pi/2\}$, leading to maximum and minimum:

$$D_{\mathrm{Cos}}(\theta = 0) = \frac{1}{\pi\alpha^2} \qquad D_{\mathrm{Cos}}\left(\theta = \frac{\pi}{2}\right) = 0. \tag{A.11}$$

A.3.5 Relation of Regularization Parameters

In Section V.4 we invert the relation between a radius r_0 and our parameter τ_0. The math behind uses only equivalent transformations as shown below:

$$\begin{aligned}
& \tau_0 = \frac{1}{2\pi(1-\cos(\arctan(r_0/d))} \\
\Leftrightarrow \quad & \frac{1}{2\pi\tau_0} = 1-\cos(\arctan(r_0/d)) \\
\Leftrightarrow \quad & 1-\frac{1}{2\pi\tau_0} = \cos(\arctan(r_0/d)) \\
\Leftrightarrow \quad & \frac{2\pi\tau_0-1}{2\pi\tau_0} = \cos(\arctan(r_0/d)) \\
\Leftrightarrow \quad & \tan\left(\arccos\left(\frac{2\pi\tau_0-1}{2\pi\tau_0}\right)\right) = \frac{r_0}{d} \\
\Leftrightarrow \quad & d\tan\left(\arccos\left(\frac{2\pi\tau_0-1}{2\pi\tau_0}\right)\right) = r_0
\end{aligned}$$

By using $\tan(\arccos(x)) = \sqrt{1-x^2}/x$ we can get rid of the trigonometric functions, too:

$$\begin{aligned}
\Leftrightarrow \quad & r_0 = d\sqrt{1-\left(\frac{2\pi\tau_0-1}{2\pi\tau_0}\right)^2}\frac{2\pi\tau_0}{2\pi\tau_0-1} \\
\Leftrightarrow \quad & r_0 = d\sqrt{\frac{4\pi^2\tau_0^2-(2\pi\tau_0-1)^2}{4\pi^2\tau_0^2}}\frac{2\pi\tau_0}{2\pi\tau_0-1} \\
\Leftrightarrow \quad & r_0 = d\sqrt{4\pi^2\tau_0^2-(2\pi\tau_0-1)^2}\frac{1}{2\pi\tau_0-1} \\
\Leftrightarrow \quad & r_0 = d\frac{\sqrt{4\pi\tau_0-1}}{2\pi\tau_0-1}
\end{aligned}$$

Next, we insert the above r_0 in Equation (II.90) with $\lambda \in \{1/6, 1/12\}$, yielding the iteration dependent parameter

$$r_i = r_0 \cdot i^{-\lambda} = d\frac{\sqrt{4\pi\tau_0-1}}{2\pi\tau_0-1}\cdot i^{-\lambda}. \tag{A.12}$$

In the last step we reinsert r_i in Equation (V.25):

$$\tau_i = \frac{1}{2\pi\left[1-\cos\left(\arctan\left(\cancel{d}\frac{\sqrt{4\pi\tau_0-1}}{2\pi\tau_0-1}\cdot i^{-\lambda}/\cancel{d}\right)\right)\right]} \tag{A.13}$$

where d gets canceled out.

A.4 Algorithms and Source Codes

A.4.1 Interpolated Octree Sampling

Section III.2.1 p. 91 briefly introduced the idea of subdividing nodes on the fly while descending the tree. While the idea is comparably simple, it is tricky to implement.

```
func sample_interpolated(DensityOctree tree, f32[3] pos) -> f32
  offPos = pos - tree.sceneMin
  normPos = offPos / tree.sceneSize
  iPos = <i32>(normPos * (1 << 31))
  # Memory to track the nodes and their areas (ping pong -> 16)
  i32[16] address = 0
  f32[16] area = -1.0 # -1 marks as not yet computed
  par = 0   # Offset into address and area buffer for parents
  cur = 8   # Offset into address and area buffer for current nodes
  lvl = 0
  # Position of the cell with the smallest index among the 8 cells
  # for interpolation on the child level (multiplied with 2).
  i32[3] minParentPos = 0
  anyHasChildren = tree.data[0] < 0
  while anyHasChildren:
    ++lvl
    swap(par, cur)
    # The next higher level helps to decide if we are left or right
    # in the current cell, which is important for interpolation.
    nextLvlPos = iPos >> (30 - lvl)
    # Get the current level grid coordinate for the cell with the
    # smallest index among the 8 cells for interpolation
    minGridPos = nextLvlPos / 2 - 1 + (nextLvlPos & 1)
    cellSize = tree.sceneSize / (1 << lvl)
    # Copy or track each of the current 8 cells
    anyHasChildren = false  # Record if any new cell has children
    for i in [0,7]:
      cellPos = minGridPos + [i&1, (i>>1)&1, i>>2]
      localParent = (cellPos - minParentPos) / 2
      parentIdx = dot(localParent, [1, 2, 4])
      # Check if parent node has children
      parentAddress = address[par+parentIdx]
      countOrChild = tree.data[parentAddress]
      if countOrChild < 0:
        # Yes, insert child node's address
        localChildIdx = dot(cellPos & 1, [1, 2, 4])
        address[cur+i] = -countOrChild + localChildIdx
        countOrChild = tree.data[i]
        if countOrChild >= 0:
          # This is a leaf, compute its area
          localPos = offPos - cellPos * cellSize
          area[cur+i] = intersection_area(cellSize, localPos, normal)
        anyHasChildren |= countOrChild < 0
      else
        # No node on current level, copy parent as virtual
        address[cur+i] = parentAddress
        area[cur+i] = area[par+parentIdx]
      end
    end
    minParentPos = minGridPos * 2
  end
  return interpolate(tree, address[cur:cur+8'], area[cur:cur+8'],
                     1 << lvl, normPos, normal)
end

func interpolate(DensityOctree tree, i32[8] address, f32[8] pArea,
                 i32 lvlRes, f32[3] normPos, f32[3] normal) -> f32
  tPos = normPos * lvlRes - 0.5
  minGridPos = floor(tPos)
```

```
  f32[3][2] w   # weights for interpolation
  w[1] = tPos - gridPos
  w[0] = 1 - w[1]
  cellSize = tree.sceneSize / lvlRes
  eps = 0.01 * dot(cellSize, cellSize[1,2,0]) / 3
  countSum = 0.0
  areaSum = 0.0
  for i in [0,7]:
    cellPos = minGridPos + [i&1, (i>>1)&1, i>>2]
    localPos = offPos - cellPos * cellSize
    area = intersection_area(cellSize, localPos, normal)
    if area > 0 and pArea > 0:
      w = ws[i&1] * ws[(i>>1)&1] * ws[i>>2]
      lvlFactor = (area + eps) / (pArea[i] + eps)
      count = tree.data[address[i]] * lvlFactor
      countSum += count * w
      areaSum += area * w
    end
  end
  return countSum / areaSum
end
```

A.4.2 COMPUTING CURVATURE

Since it is not trivial to obtain the mean surface curvature at a hit point, this section gives a brief description of the algorithm I used. There are several approaches to compute the curvature on triangle meshes [Goldfeather and Interrante 2004; Rusinkiewicz 2004; Surazhsky et al. 2003; Taubin 1995] of which I tried various ones. Finally, I ended up with a modified *Normal Curvature Fit* method, since it works well for polygonal meshes with triangles and quads mixed.

Fundamentally, the curvature of a point on a 2D manifold is characterized by the Weingarten matrix

$$\boldsymbol{W} = \begin{bmatrix} \frac{eG-fF}{EG-F^2} & \frac{fE-eF}{EG-F^2} \\ \frac{fG-gF}{EG-F^2} & \frac{gE-fF}{EG-F^2} \end{bmatrix} \tag{A.14}$$

where

$$\begin{aligned} E &= \left\langle \frac{\partial \boldsymbol{x}}{\partial u}, \frac{\partial \boldsymbol{x}}{\partial u} \right\rangle & e &= \left\langle \boldsymbol{n}, \frac{\partial^2 \boldsymbol{x}}{\partial u^2} \right\rangle \\ F &= \left\langle \frac{\partial \boldsymbol{x}}{\partial v}, \frac{\partial \boldsymbol{x}}{\partial u} \right\rangle & f &= \left\langle \boldsymbol{n}, \frac{\partial^2 \boldsymbol{x}}{\partial u \partial v} \right\rangle \\ G &= \left\langle \frac{\partial \boldsymbol{x}}{\partial v}, \frac{\partial \boldsymbol{x}}{\partial v} \right\rangle & g &= \left\langle \boldsymbol{n}, \frac{\partial^2 \boldsymbol{x}}{\partial v^2} \right\rangle . \end{aligned}$$

for some tangent vectors $\boldsymbol{u} = {}^{\partial \boldsymbol{x}}/_{\partial u}, \boldsymbol{v} = {}^{\partial \boldsymbol{x}}/_{\partial v} \in \mathbb{R}^3$. Note that it is possible to choose orthonormal tangent vectors $\boldsymbol{u}, \boldsymbol{v}$ such that $E = G = 1$ and $F = 0$. Thus the matrix can be simplified to

$$\boldsymbol{W} = \begin{bmatrix} e & f \\ f & g \end{bmatrix}$$

if we choose the tangents properly.

The Weingarten matrix tells us how large the curvature in a given unit direction $\boldsymbol{y}$ in tangent space is, namely $\kappa_{\boldsymbol{y}} = \left\| \boldsymbol{y}^T \boldsymbol{W} \boldsymbol{y} \right\|$. The Gaussian curvature, which is the product of the two principal curvatures, can be computed with

$$K = \kappa_1 \cdot \kappa_2 = \frac{eg - f^2}{EG - F^2} = eg - f^2 \tag{A.15}$$

and finally the mean curvature, in which we are interested, is computed by

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{1}{2} \cdot \frac{eG + gE - 2fF}{EG - F^2} = e + g. \tag{A.16}$$

The idea of *Normal Curvature Fit* methods is to compute the curvature κ_i along each projected, normalized edge $\boldsymbol{y}_i$ adjacent to a vertex. The vertex's curvature is then a fit of the equations

$$\begin{aligned} & \boldsymbol{y}_i^T \boldsymbol{W} \boldsymbol{y}_i = \kappa_i \\ \Leftrightarrow \quad & (u_i,\ v_i) \begin{bmatrix} e & f \\ f & g \end{bmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \kappa_i \\ \Leftrightarrow \quad & (u_i^2,\ 2u_i v_i,\ v_i^2) \begin{pmatrix} e \\ f \\ g \end{pmatrix} = \kappa_i \end{aligned} \tag{A.17}$$

over all adjacent edges. That means each of the edges defines one equation with three unknowns e, f and g. This leads to an overdetermined equation system for most configurations (valence greater two), which is solved in the least squares sense to obtain the final values of the vertex's tensor.

If we only want to know the principal curvature values κ_1 and κ_2, we can simplify the equations by setting $f = 0$. This fixes the rotation of the tangent frame such that we are not able to tell the directions of principal curvature. In consequence each edge yields an equation $u_i^2\kappa_1 + v_i^2\kappa_2 = \kappa_i$. This is a direct application of Euler's theorem

$$\cos^2\theta_i \cdot \kappa_1 + \sin^2\theta_i \cdot \kappa_2 = \kappa_i \tag{A.18}$$

because the vector $\boldsymbol{y}^T = (u_i, v_i)$ is normalized and equals $(\cos\theta_i, \sin\theta_i)$.

Thus, to compute H it suffices to solve an equation system with only two unknowns, which is well defined if the vertex has a valence of two or greater and is more robust than the three variables fit.

The algorithm to compute the vertex curvatures works as following. In the first step we choose the tangent vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ orthonormal to the vertex normal $\boldsymbol{n}$ and to each other. Then, we can simply compute the projected edge $\boldsymbol{y}_i = (u_i\ v_i)^T$ via

$$u_i = \frac{\langle \boldsymbol{e}_i, \boldsymbol{u}\rangle}{\langle \boldsymbol{e}_i, \boldsymbol{u}\rangle^2 + \langle \boldsymbol{e}_i, \boldsymbol{v}\rangle^2} \tag{A.19a}$$

$$v_i = \frac{\langle \boldsymbol{e}_i, \boldsymbol{v}\rangle}{\langle \boldsymbol{e}_i, \boldsymbol{u}\rangle^2 + \langle \boldsymbol{e}_i, \boldsymbol{v}\rangle^2} \tag{A.19b}$$

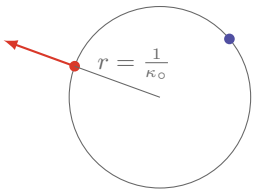
$$\text{with} \quad \boldsymbol{e}_i = \boldsymbol{x} - \boldsymbol{x}_i$$

for the vertex position $\boldsymbol{x}$ and all vertices $\boldsymbol{x}_i$ in the 1-neighborhood of this vertex.

Then, the edge curvature κ_i is computed, where several options are available. Finally, the resulting equations over all edges are solved in least squares sense $\boldsymbol{A}^T\boldsymbol{A}\,(\kappa_1, \kappa_2)^T = \boldsymbol{A}^T\boldsymbol{\kappa}$, where $\boldsymbol{\kappa}$ is the vector of all κ_i. It is possible to update $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{\kappa}$ iteratively, such that it is not necessary to store the variable number of equations.

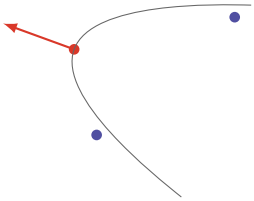
Computing the curvature κ_i of an edge is more difficult. It must be approximated under some basic assumption. The results under different common assumptions are:

Circular fit The curvature of the unique circle passing through $\boldsymbol{x}$ and $\boldsymbol{x_i}$ with normal $\boldsymbol{n}$ at $\boldsymbol{x}$ is



$$\kappa_{\circ,i} = 2\frac{\langle \boldsymbol{e}_i, \boldsymbol{n}\rangle}{\langle \boldsymbol{e}_i, \boldsymbol{e}_i\rangle} \tag{A.20}$$

Paraboloid fit In this method a paraboloid with apex at $\boldsymbol{x}$ and normal $\boldsymbol{n}$ is fitted to all neighbored vertices $\boldsymbol{x}_i$. This also yields a least squares solution, where the equations are identical to the normal fit method with



$$\kappa_{\wedge,i} = 2\frac{\langle \boldsymbol{e}_i, \boldsymbol{n}\rangle}{\langle \boldsymbol{e}_i, \boldsymbol{u}\rangle^2 + \langle \boldsymbol{e}_i, \boldsymbol{v}\rangle^2} \tag{A.21}$$

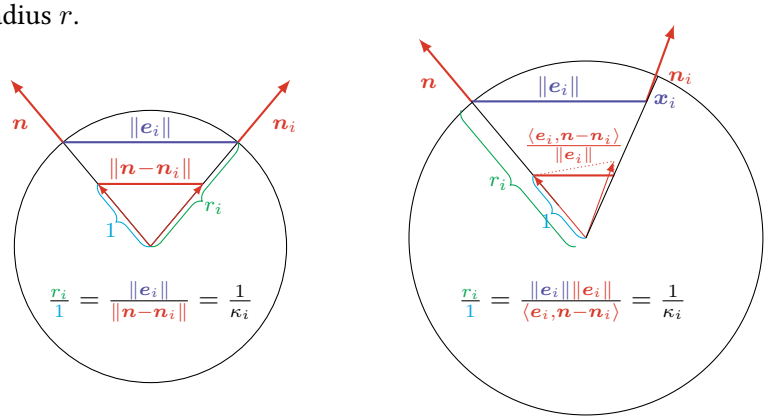
as was shown by Goldfeather and Interrante [2004].

Cubic fit Goldfeather and Interrante [2004] fitted a higher order surface to make use of the normals $\boldsymbol{n}_i$ at the neighbor vertices. This reduces noise and parametrization issues which are apparent in the two above methods. However, the cubic fit triples the number of equations and requires more computations.

Projected circular fit Another idea I had not seen in the literature, is to include both normals $\boldsymbol{n}$ and $\boldsymbol{n_i}$ through a projection

$$\kappa_{\perp,i} = \frac{\langle \boldsymbol{e}_i, \boldsymbol{n} - \boldsymbol{n}_i \rangle}{\langle \boldsymbol{e}_i, \boldsymbol{e}_i \rangle}. \tag{A.22}$$

The idea is to apply the intercept theorem in a circle to compute its radius r.



This can be generalized to non-circular normals by taking the projection along the edge direction. As visible on the right, this approach fits a circle which does not pass through $\boldsymbol{x}_i$, but partially involves $\boldsymbol{n}_i$.

Projected paraboloid fit Several works have compared the quality of the different curvature approximations [Goldfeather and Interrante 2004; Surazhsky et al. 2003]. They found that the *paraboloid fit* worked better than the *circular fit* for computing the mean curvature H. Comparing Equations (A.20) and (A.21), the only difference is the denominator. Hence, deriving a similar form of Equation (A.22) seems very reasonable:

$$\kappa_{\perp\!\!\!\perp i} = \frac{\langle \boldsymbol{e}_i, \boldsymbol{n} - \boldsymbol{n}_i \rangle}{\langle \boldsymbol{e}_i, \boldsymbol{u} \rangle^2 + \langle \boldsymbol{e}_i, \boldsymbol{v} \rangle^2}. \tag{A.23}$$

Figure A.2 visually compares four of the methods. Both, circle and paraboloid fitting, produce artifacts dependent on the tessellation (see closeups). The projected fit methods smooth this artifact and other regions of the mesh. Both parabolic approaches produce slightly larger values, which is closer to the ground truth on average [Surazhsky et al. 2003].

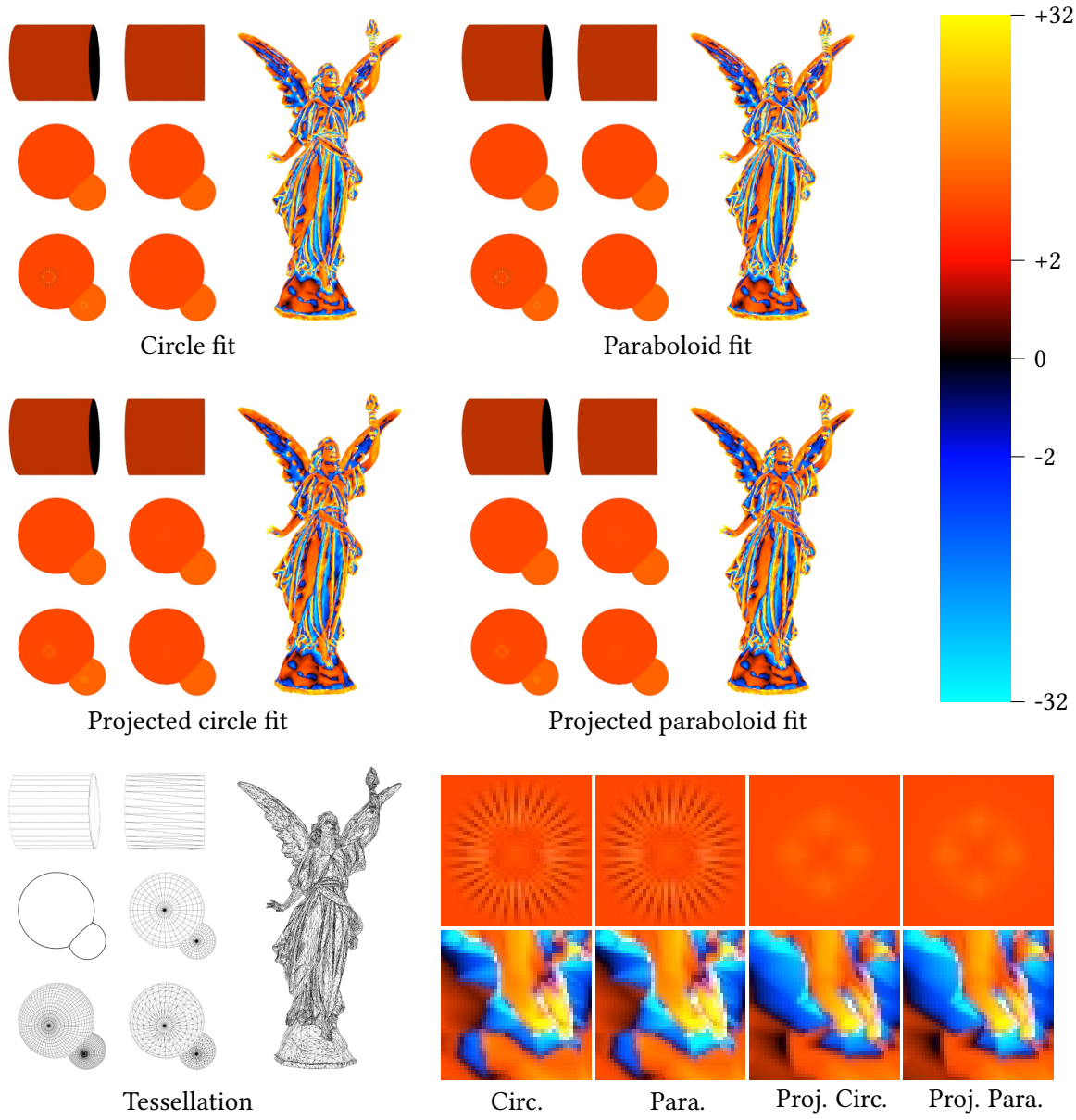


Figure A.2: Comparison of the mean curvature value using different methods to compute the edge curvature κ. Note the artifact on the bottom left spheres (also shown in the closeup), which is smoothed under the projected circular fit method.

The following code snipped shows the final implementation of the most successful method.

```
func compute_pervertex_curvature(Mesh m)
  for v in m.vertices:
    u = perpendicular(v.normal)
    v = cross(v.normal, u)
    f32[2][2] AtA = 0
    f32[2] Atk = 0
    for vi in v.neighborhood():
      ei = v.position - vi.position
      y = [dot(u, ei), dot(v, ei)]
      yNormSq = dot(y,y) + 1e-30
      # Projected paraboloid fit
      ki = dot(v.normal - vi.normal, ei) / yNormSq
      # Here you could alternatively compute:
      # eiNormSq = dot(ei, ei) + 1e-30
      # Circular fit: ki = 2 * dot(v.normal, ei) / eiNormSq
      # Proj Circular: ki = dot(v.normal - vi.normal, ei) / eiNormSq
      # Paraboloid fit: ki = 2 * dot(v.normal, ei) / yNormSq
      uiSq = y[0] * y[0] / yNormSq
      viSq = y[1] * y[1] / yNormSq
      # Update equation system
      Atk += ki * [uiSq, viSq]
      AtA += [[uiSq * uiSq, uiSq * viSq],
              [uiSq * viSq, viSq * viSq]]
    end
    # Solve the 2x2 least squares system with Cramer's rule
    detA = det(AtA) + 1e-30
    kappa1 = (Atk[0] * AtA[1][1] - Atk[1] * AtA[0][1]) / detA
    kappa2 = (Atk[1] * AtA[0][0] - Atk[0] * AtA[1][0]) / detA
    v.H = (kappa1 + kappa2) / 2
  end
end
```

Rusinkiewicz [2004] proposed a method which first computes a tensor per face and then averages these at the vertices. It is in general more robust with respect to tessellation than the above fitting methods. However, it is also more expensive to compute and needs more temporary memory.

Bibliography

Aila, Timo and **Laine**, Samuli (2009).
Understanding the Efficiency of Ray Traversal on GPUs
In: *Proc. of High Performance Graphics (TOG)*. HPG, pp. 145–149.
DOI: 10.1145/1572769.1572792.

Akenine-Möller, Tomas; **Nilsson**, Jim; **Andersson**, Magnus; **Barré-Brisebois**, Colin; **Toth**, Robert and **Karras**, Tero (2019).
Texture Level of Detail Strategies for Real-Time Ray Tracing
In: *Ray Tracing Gems*. Ed. by Eric **Haines** and Tomas **Akenine-Möller**. 1st ed., pp. 321–345.
http://raytracinggems.com.

Alcantara, Dan Anthony Feliciano (2011).
Efficient Hash Tables on the GPU
PhD thesis. University of California at Davis.
http://idav.ucdavis.edu/~dfalcant/research.php.

Arvo, James (1986).
Backward Ray Tracing
In: *Developments in Ray Tracing (SIGGRAPH Course Notes)*. Vol. 12, pp. 259–263.

Special Issue On Production Rendering and Regular Papers (2018)
In: *ACM Transactions on Graphics (TOG)* 37.3. Ed. by Kavita **Bala**. ISSN: 0730-0301.

Beckmann, Petr and **Spizzichino**, André (1963).
The Scattering of Electromagnetic Waves from Rough Surfaces
International Series of Monographs on Electromagnetic Waves. vol. 4.

Beer, August (1852).
Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten
In: *Annalen der Physik und Chemie*.

Bekaert, Philippe; **Slusallek**, Philipp; **Cools**, Ronald; **Havran**, Vlastimil and **Seidel**, Hans-Peter (2003).
A Custom Designed Density Estimator for Light Transport
Research Report MPI-I-2003-4-004. Saarbrücken: Max-Planck Institut für Informatik.
http://domino.mpi-inf.mpg.de/internet/reports.nsf/NumberView/2003-4-004.

Belcour, Laurent; **Soler**, Cyril; **Subr**, Kartic; **Holzschuch**, Nicolas and **Durand**, Fredo (2013).
5D Covariance Tracing for Efficient Defocus and Motion Blur
In: *ACM Transaction on Graphics (TOG)* 32.3, 31:1–31:18.
DOI: 10.1145/2487228.2487239.
https://maverick.inria.fr/Members/Laurent.Belcour/covariance-tracing/.

Bentley, Jon Louis (1975).
Multidimensional Binary Search Trees Used for Associative Searching
In: *Communications of the ACM* 18.9, pp. 509–517.
DOI: 10.1145/361002.361007.

Bhatia, Rajendra and **Davis**, Chandler (2000).
A Better Bound on the Variance
In: *The American Mathematical Monthly* 107.4, pp. 353–357.
DOI: 10.2307/2589180.
http://www.jstor.org/stable/2589180.

Blackman, David and **Vigna**, Sebastiano (2018).
Scrambled Linear Pseudorandom Number Generators
arXiv:1805.01407.
http://arxiv.org/abs/1805.01407.

Blinn, James F. and **Newell**, Martin E. (1976).
Texture and Reflection in Computer Generated Images
In: *Communications of the ACM* 19.10, pp. 542–547.
DOI: 10.1145/360349.360353.

Bouchard, Guillaume; **Iehl**, Jean-Claude; **Ostromoukhov**, Victor and **Poulin**, Pierre (2013).
Improving Robustness of Monte-Carlo Global Illumination with Directional Regularization
In: *SIGGRAPH Asia 2013 Technical Briefs*. SA '13, 22:1–22:4.
DOI: 10.1145/2542355.2542383.

Box, George Edward Pelham and **Muller**, Mervin Edgar (1958).
A Note on the Generation of Random Normal Deviates
In: *The Annals of Mathematical Statistics* 29.2, pp. 610–611.
https://www.jstor.org/stable/2237361.

Bromiley, Paul (2003).
Products and Convolutions of Gaussian Probability Density Functions
In: *Tina-Vision Memo* 3.

Burley, Brent; **Adler**, David; **Chiang**, Matt Jen-Yuan; **Driskill**, Hank; **Habel**, Ralf; **Kelly**, Patrick; **Kutz**, Peter; **Li**, Yining Karl and **Teece**, Daniel (2018).
The Design and Evolution of Disney's Hyperion Renderer
In: *ACM Transaction on Graphics (TOG)* 37.3, 33:1–33:22.
DOI: 10.1145/3182159.

Chaitanya, Chakravarty Reddy Alla; **Belcour**, Laurent; **Hachisuka**, Toshiya; **Premoze**, Simon; **Pantaleoni**, Jacopo and **Nowrouzezahrai**, Derek (2018).
Matrix Bidirectional Path Tracing
In: *Proc. of Eurographics Symposium on Rendering (EGSR)*, pp. 023–032.
DOI: 10.2312/sre.20181169.

Chiang, Matt Jen-Yuan; **Bitterli**, Benedikt; **Tappan**, Chuck and **Burley**, Brent (2016).
A Practical and Controllable Hair and Fur Model for Production Path Tracing
In: *Computer Graphics Forum (CGF)* 35.2, pp. 275–283.
DOI: 10.1111/cgf.12830.

Christensen, Per H. and **Batali**, Dana (2004).
An Irradiance Atlas for Global Illumination in Complex Production Scenes
In: *Rendering Techniques '04 (Proc. EGSR)*. Vol. 2004. EGSR, pp. 133–141.
DOI: 10.2312/EGWR/EGSR04/133-141.

Christensen, Per H.; **Kensler**, Andrew and **Kilpatrick**, Charlie (2018).
Progressive Multi-Jittered Sample Sequences
In: *Computer Graphics Forum (CGF)* 37.4, pp. 21–33.
DOI: 10.1111/cgf.13472.

Christensen, Per H.; **Laur**, David M.; **Fong**, Julia; **Wooten**, Wayne L. and **Batali**, Dana (2003).

Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes
In: *Computer Graphics Forum (CGF)* 22.3, pp. 543–552.
DOI: 10.1111/1467-8659.t01-1-00702.
https://www.seanet.com/~myandper/abstract/eg03.htm.

Christensen, Per; **Fong**, Julian; **Shade**, Jonathan; **Wooten**, Wayne; **Schubert**, Brenden; **Kensler**, Andrew; **Friedman**, Stephen; **Kilpatrick**, Charlie; **Ramshaw**, Cliff; **Bannister**, Marc; **Rayner**, Brenton; **Brouillat**, Jonathan and **Liani**, Max (2018).
RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering
In: *ACM Transaction on Graphics (TOG)* 37.3, 30:1–30:21.
DOI: 10.1145/3182162.

Cohen, Michael F.; **Chen**, Shenchang Eric; **Wallace**, John R. and **Greenberg**, Donald P (1988).
A Progressive Refinement Approach to Fast Radiosity Image Generation
In: *Computer Graphics (Proc. ACM SIGGRAPH)* 22, pp. 75–84.
DOI: 10.1145/378456.378487.

Crassin, Cyril (2011).
GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes
PhD thesis. Universite de Grenoble.
http://maverick.inria.fr/Publications/2011/Cra11.

d'Eon, Eugene and **Luebke**, David (2007).
Advanced Techniques for Realistic Real-Time Skin Rendering
In: *GPU Gems 3*. 2nd ed.
https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch14.html.

Dachsbacher, Carsten; **Křivánek**, Jaroslav; **Hašan**, Miloš; **Arbree**, Adam; **Walter**, Bruce and **Novák**, Jan (2014).
Scalable Realistic Rendering with Many-Light Methods
In: *Computer Graphics Forum (CGF)* 33.1, pp. 88–104.
DOI: 10.1111/cgf.12256.
http://drz.disneyresearch.com/~jnovak/publications/ManyLightSTAR/index.html.

Davidovič, Tomáš; **Křivánek**, Jaroslav; **Hašan**, Miloš and **Slusallek**, Philipp (2014).
Progressive Light Transport Simulation on the GPU: Survey and Improvements
In: *ACM Transactions on Graphics (TOG)* 33.3, p. 29.
DOI: 10.1145/2602144.

Dupuy, Jonathan; **Heitz**, Eric and **Belcour**, Laurent (2017).
A Spherical Cap Preserving Parameterization for Spherical Distributions
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36.4, 139:1–139:12.
DOI: 10.1145/3072959.3073694.

Dupuy, Jonathan; **Heitz**, Eric; **Iehl**, Jean-Claude; **Poulin**, Pierre; **Neyret**, Fabrice and **Ostromoukhov**, Victor (2013).
Linear Efficient Antialiased Displacement and Reflectance Mapping
In: *ACM Transaction on Graphics (TOG)* 32.6, 211:1–211:11.
DOI: 10.1145/2508363.2508422.

Dutré, Philip; **Lafortune**, Eric P. and **Willems**, Yves D. (1993).
Monte Carlo Light Tracing with Direct Computation of Pixel Intensities
In: *Proc. of Computational Graphics and Visualisation Techniques*, pp. 128–137.
http://graphics.cs.kuleuven.be/publications/MCLTWDCOPI/.

Elvira, Víctor; **Martino**, Luca; **Luengo**, David and **Bugallo**, Mónica F (2017).
Generalized Multiple Importance Sampling
In: *arXiv:1511.03095v2*.
https://arxiv.org/abs/1511.03095v2.

Evans, Glenn F. and **McCool**, Micheal D. (1999).
Stratified Wavelength Clusters for Efficient Spectral Monte Carlo Rendering
In: *Proc. of Graphics Interface*, pp. 42–49.
http://dl.acm.org/citation.cfm?id=351631.351648.

Fascione, Luca; **Hanika**, Johannes; **Fajardo**, Marcos; **Christensen**, Per; **Burley**, Brent and **Green**, Brian (2017a).
Path Tracing in Production - Part 1: Production Renderers
In: *ACM SIGGRAPH Courses*, 13:1–13:39.
DOI: 10.1145/3084873.3084904.

Fascione, Luca; **Hanika**, Johannes; **Pieké**, Rob; **Hery**, Christophe; **Villemin**, Ryusuke; **Schmidt**, Thorsten-Walther; **Kulla**, Christopher; **Heckenberg**, Daniel and **Mazzone**, André (2017b).
Path Tracing in Production - Part 2: Making Movies
In: *ACM SIGGRAPH Courses*, 15:1–15:32.
DOI: 10.1145/3084873.3084906.

Fechner, Gustav Theodor (1858).
Über ein wichtiges psychophysisches Grundgesetz und dessen Beziehung zur Schätzung der Sterngrössen
In: *Ges. Wissensch., Math.-Phys. Kl.*

Fong, Julian; **Wrenninge**, Magnus; **Kulla**, Christopher and **Habel**, Ralf (2017).
Production Volume Rendering
In: *ACM SIGGRAPH Courses*, 2:1–2:79.
DOI: 10.1145/3084873.3084907.
https://graphics.pixar.com/library/ProductionVolumeRendering/.

García, R.; **Ureña**, C. and **Sbert**, M. (2012).
Description and Solution of an Unreported Intrinsic Bias in Photon Mapping Density Estimation with Constant Kernel
In: *Computer Graphics Forum* 31.1, pp. 33–41.
DOI: 10.1111/j.1467-8659.2011.02081.x.

Georgiev, Iliyan; **Ize**, Thiago; **Farnsworth**, Mike; **Montoya-Vozmediano**, Ramón; **King**, Alan; **Lommel**, Brecht Van; **Jimenez**, Angel; **Anson**, Oscar; **Ogaki**, Shinji; **Johnston**, Eric; **Herubel**, Adrien; **Russell**, Declan; **Servant**, Frédéric and **Fajardo**, Marcos (2018).
Arnold: A Brute-Force Production Path Tracer
In: *ACM Transaction on Graphics (TOG)* 37.3, 32:1–32:12.
DOI: 10.1145/3182160.

Georgiev, Iliyan; **Křivánek**, Jaroslav; **Davidovič**, Tomáš and **Slusallek**, Philipp (2012).
Light Transport Simulation with Vertex Connection and Merging
In: *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 31.6, 192:1–192:10.
DOI: 10.1145/2366145.2366211.
https://cgg.mff.cuni.cz/~jaroslav/papers/2012-vcm/.

Goldfeather, Jack and **Interrante**, Victoria (2004).
A Novel Cubic-order Algorithm for Approximating Principal Direction Vectors
In: *ACM Transaction on Graphics (TOG)* 23.1, pp. 45–63.
DOI: 10.1145/966131.966134.

Greenberg, Donald P.; **Cohen**, Michael F. and **Torrance**, Kenneth E. (1986).
Radiosity: A Method for Computing Global Illumination
In: *The Visual Computer* 2.5, pp. 291–297.
DOI: 10.1007/BF02020429.

Gu, Feng; **Jendersie**, Johannes and **Grosch**, Thorsten (2018).
Fast and Dynamic Construction of Bounding Volume Hierarchies based on Loose Octrees
In: *Proc. Vision, Modeling and Visualization (VMV)*.
DOI: 10.2312/vmv.20181257.

Guild, John et al. (1931).
The Colorimetric Properties of the Spectrum
In: *Philosophical Transactions of the Royal Society of London (A)* 230.681, pp. 149–187.
http://rsta.royalsocietypublishing.org/content/230/681-693/149.

Günther, Johannes; **Chen**, Tongbo; **Goesele**, Michael; **Wald**, Ingo and **Seidel**, Hans-Peter (2005).
Efficient Acquisition and Realistic Rendering of Car Paint
In: *Proc. of Vision, Modeling, and Visualization (VMV)*. ISBN: 3-89838-068-8.
http://www.johannes-guenther.net/carpaint/.

Hachisuka, Toshiya and **Jensen**, Henrik Wann (2009).
Stochastic Progressive Photon Mapping
In: *ACM Transactions on Graphics (TOG)* 28.5, 141:1–141:8.
DOI: 10.1145/1618452.1618487.

– (2010).
Parallel Progressive Photon Mapping on GPUs
In: *ACM SIGGRAPH Asia Sketches*, p. 54.
DOI: 10.1145/1899950.1900004.

Hachisuka, Toshiya; **Ogaki**, Shinji and **Jensen**, Henrik Wann (2008).
Progressive Photon Mapping
In: *ACM Transactions on Graphics (TOG)* 27 number 5, p. 130.
DOI: 10.1145/1409060.1409083.

Hachisuka, Toshiya; **Pantaleoni**, Jacopo and **Jensen**, Henrik Wann (2012).
A Path Space Extension for Robust Light Transport Simulation
In: *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 31.6, 191:1–191:10.
DOI: 10.1145/2366145.2366210.

Hanika, Johannes; **Droske**, Marc and **Fascione**, Luca (2015a).
Manifold Next Event Estimation
In: *Computer Graphics Forum (Proc. EGSR)* 34.4, pp. 87–97.
DOI: 10.1111/cgf.12681.

Hanika, Johannes; **Kaplanyan**, Anton and **Dachsbacher**, Carsten (2015b).
Improved Half Vector Space Light Transport
In: *Computer Graphics Forum* 34.4, pp. 65–74.
DOI: 10.1111/cgf.12679.

Havran, Vlastimil; **Herzog**, Robert and **Seidel**, Hans-Peter (2005).
Fast Final Gathering via Reverse Photon Mapping
In: *Computer Graphics Forum (CGF)* 24.3, pp. 323–332.
DOI: 10.1111/j.1467-8659.2005.00857.x.

Heckbert, Paul S. (1990).
Adaptive Radiosity Textures for Bidirectional Ray Tracing
In: *Computer Graphics (Proc. ACM SIGGRAPH)* 24.4, pp. 145–154.
DOI: 10.1145/97880.97895.

Heitz, Eric (2014).
Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs
In: *Journal of Computer Graphics Techniques (JCGT)* 3.2, pp. 48–107.
http://jcgt.org/published/0003/02/03/.

– (2019).
A Low-Distortion Map Between Triangle and Square
https://hal.archives-ouvertes.fr/hal-02073696.

Heitz, Eric and **d'Eon**, Eugene (2014).
Importance Sampling Microfacet-Based BSDFs Using the Distribution of Visible Normals
In: *Computer Graphics Forum (CGF)* 33.4, pp. 103–112.
DOI: 10.1111/cgf.12417.

Heitz, Eric; **Dupuy**, Jonathan; **Hill**, Stephen and **Neubelt**, David (2016a).
Real-time Polygonal-light Shading with Linearly Transformed Cosines
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35.4, 41:1–41:8.
DOI: 10.1145/2897824.2925895.

Heitz, Eric; **Hanika**, Johannes; **d'Eon**, Eugene and **Dachsbacher**, Carsten (2016b).
Multiple-scattering Microfacet BSDFs with the Smith Model
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35.4, 58:1–58:14.
DOI: 10.1145/2897824.2925943.

Helmholtz, Hermann L. F. (1867).
Handbuch der physiologischen Optik.

Henrich, Niklas; **Baerz**, Jakob; **Grosch**, Thorsten and **Müller**, Stefan (2011).
Accelerating Path Tracing by Eye-Path Reprojection
In: *International Congress on Graphics and Virtual Reality (GRVR)*.
http://www.rendering.ovgu.de/rendering_media/downloads/publications.

Herholz, Sebastian; **Elek**, Oskar; **Vorba**, Jiří; **Lensch**, Hendrik and **Křivánek**, Jaroslav (2016).
Product Importance Sampling for Light Transport Path Guiding
In: *Computer Graphics Forum (CGF)* 35.4, pp. 67–77.
DOI: 10.1111/cgf.12950.

Hernandez, R. J. Garcia; **Ureña**, Carlos; **Poch**, Jordi and **Sbert**, Mateu (2014).
Overestimation and Underestimation Biases in Photon Mapping with Non-Constant Kernels
In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20.10, pp. 1441–1450.
DOI: 10.1109/TVCG.2014.2314665.

Hickernell, Fred J.; **Lemieux**, Christiane; **Owen**, Art B. and **L'Ecuyer**, Pierre (2005).
Control Variates for Quasi-Monte Carlo
In: *Statistical Science* 20.1, pp. 1–31.
DOI: 10.2307/20061157.
http://www.jstor.org/stable/20061157.

Igehy, Homan (1999).
Tracing Ray Differentials
In: *Proceedings of SIGGRAPH*, pp. 179–186.

Immel, David S.; **Cohen**, Michael F. and **Greenberg**, Donald P. (1986).
A Radiosity Method for Non-diffuse Environments
In: *Computer Graphics (Proc. ACM SIGGRAPH)* 20.4, pp. 133–142.
DOI: 10.1145/15886.15901.

Jakob, Wenzel; **Arbree**, Adam; **Moon**, Jonathan T.; **Bala**, Kavita and **Marschner**, Steve (2010).
A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29.4, 53:1–53:13.
DOI: 10.1145/1778765.1778790.

Jakob, Wenzel and **Marschner**, Steve (2012).
Manifold Exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31.4, 58:1–58:13.
DOI: 10.1145/2185520.2185554.
http://www.cs.cornell.edu/projects/manifolds-sg12/.

Jendersie, Johannes (2018).
Path Throughput Importance Weights
arXiv:1806.01005.
https://arxiv.org/abs/1806.01005.

– (2019a).
Next Event Backtracking
arXiv:1909.00573.
https://arxiv.org/abs/1909.00573.

– (2019b).
Variance Reduction via Footprint Estimation in the Presence of Path Reuse
In: *Ray Tracing Gems*. Ed. by Eric **Haines** and Tomas **Akenine-Möller**. 1st ed., pp. 557–569.
http://raytracinggems.com.

Jendersie, Johannes and **Grosch**, Thorsten (2018).
An Improved Multiple Importance Sampling Heuristic for Density Estimates in Light Transport Simulations
In: *Proc. of Eurographics Symposium on Rendering EI&I Track (EGSR)*, pp. 65–72.
DOI: 10.2312/sre.20181173.

– (2019).
Microfacet Model Regularization for Robust Light Transport
In: *Computer Graphics Forum (Proc. of EGSR)* 38.4, pp. 39–47.
DOI: 10.1111/cgf.13768.

Jendersie, Johannes; **Kuri**, David and **Grosch**, Thorsten (2016a).
Precomputed Illuminance Composition for Real-time Global Illumination
In: *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. I3D, pp. 129–137.
DOI: 10.1145/2856400.2856407.

– (2016b).
Real-Time Global Illumination Using Precomputed Illuminance Composition with Chrominance Compression
In: *Journal of Computer Graphics Techniques (JCGT)* 5.4, pp. 8–35. ISSN: 2331-7418.
http://jcgt.org/published/0005/04/02/.

Jendersie, Johannes; **Rohmer**, Kai; **Brüll**, Felix and **Grosch**, Thorsten (2017).
Pixel Cache Light Tracing
In: *Proc. Vision, Modeling and Visualization (VMV)*, pp. 137–144.
DOI: 10.2312/vmv.20171269.

Jensen, Henrik Wann (1996).
Global Illumination using Photon Maps
In: *Proc. of Eurographics Workshop on Rendering (EGWR)*, pp. 21–30.
http://dl.acm.org/citation.cfm?id=275458.275461.

Jensen, Henrik Wann; **Marschner**, Stephen R.; **Levoy**, Marc and **Hanrahan**, Pat (2001).
A Practical Model for Subsurface Light Transport
In: *Proc. of SIGGRAPH '01*. SIGGRAPH '01, pp. 511–518.
DOI: 10.1145/383259.383319.

Kahn, H. and **Marshall**, A. W. (1953).
Methods of Reducing Sample Size in Monte Carlo Computations
In: *Journal of the Operations Research Society of America* 1.5, pp. 263–278.
DOI: 10.2307/166789.
http://www.jstor.org/stable/166789.

Kajiya, James T. (1986).
The Rendering Equation
In: *Computer Graphics (Proc. ACM SIGGRAPH)* 20.4, pp. 143–150.
DOI: 10.1145/15886.15902.

Kang, Chun-meng; **Wang**, Lu; **Xu**, Yan-ning and **Meng**, Xiang-xu (2016).
A survey of photon mapping state-of-the-art research and future challenges
In: *Frontiers of Information Technology & Electronic Engineering* 17.3, pp. 185–199.
DOI: 10.1631/FITEE.1500251.

Kaplanyan, Anton S. and **Dachsbacher**, Carsten (2013a).
Adaptive Progressive Photon Mapping
In: *ACM Transactions on Graphics (TOG)* 32.2, 16:1–16:13.
DOI: 10.1145/2451236.2451242.

– (2013b).
Path Space Regularization for Holistic and Robust Light Transport
In: *Computer Graphics Forum (CGF)* 32, pp. 63–72.
DOI: 10.1111/cgf.12026.

Kaplanyan, Anton S.; **Hanika**, Johannes and **Dachsbacher**, Carsten (2014).
The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation
In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33.4.
DOI: 10.1145/2601097.2601108.

Karras, Tero (2012).
Maximizing Parallelism in the Construction of BVHs, Octrees, and K-d Trees
In: *Proc. of High Performance Graphics (HPG)*, pp. 33–37.
DOI: 10.2312/EGGH/HPG12/033-037.

Kelemen, Csaba and **Szirmay-Kalos**, Laszlo (2001).
A Microfacet Based Coupled Specular-Matte BRDF Model with Importance Sampling
In: *Proc. of Eurographics Short Presentations*. Vol. 2, p. 4.
http://sirkan.iit.bme.hu/~szirmay/scook_link.htm.

Kelemen, Csaba; **Szirmay-Kalos**, László; **Antal**, György and **Csonka**, Ferenc (2002).
A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm
In: *Computer Graphics Forum (CGF)* 21, pp. 531–540.
DOI: 10.1111/1467-8659.t01-1-00703.

Keller, Alexander (1997).
Instant Radiosity
In: *Proceedings of SIGGRAPH '97*. SIGGRAPH, pp. 49–56.
DOI: 10.1145/258734.258769.

Keller, Alexander and **Wald**, Ingo (2000).
Efficient Importance Sampling Techniques for the Photon Map
In: *Proc. of Vision, Modeling, and Visualization (VMV)*, pp. 271–278.

Khungurn, Pramook and **Marschner**, Steve (2017).
Azimuthal Scattering from Elliptical Hair Fibers
In: *ACM Transaction on Graphics (TOG)* 36.2, 13:1–13:23.
DOI: 10.1145/2998578.

Knaus, Claude and **Zwicker**, Matthias (2011).
Progressive Photon Mapping: A Probabilistic Approach
In: *ACM Transactions on Graphics (TOG)* 30.3, p. 25.
DOI: 10.1145/1966394.1966404.

Kondapaneni, Ivo; **Vévoda**, Petr; **Grittmann**, Pascal; **Skřivan**, Tomáš; **Slusallek**, Philipp and **Křivánek**, Jaroslav (2019).
Optimal Multiple Importance Sampling
In: *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 38.4, 37:1–37:14.
DOI: 10.1145/3306346.3323009.

Kuipers, Lauwerens and **Niederreiter**, Harald (1974).
Uniform Distribution of Sequences
ISBN: 0-471-51045-9.

Kulla, Christopher and **Estevez**, Alejandro Conty (2017).
Revisiting Physically Based Shading at Imageworks
In: *ACM SIGGRAPH Courses*.

Lafortune, Eric P. and **Willems**, Yves D. (1993).
Bi-Directional Path Tracing
In: *Proc. of Conference on Computational Graphics and Visualization Techniques*, pp. 145–153.
https://lirias.kuleuven.be/handle/123456789/132773.

Laine, Samuli; **Saransaari**, Hannu; **Kontkanen**, Janne; **Lehtinen**, Jaakko and **Aila**, Timo (2007).
Incremental Instant Radiosity for Real-time Indirect Illumination
In: *Proc. of Eurographics Symposium on Rendering (EGSR)*. EGSR, pp. 277–286.
DOI: 10.2312/EGWR/EGSR07/277-286.

Lambert, Johann Heinrich (1760).
Photometrie. Photometria sive de mesura et gradibus luminis, colorum et umbrae
German translation 1892.
https://archive.org/details/bub_gb_zmpJAAAAYAAJ.

Malvar, Henrique and **Sullivan**, Gary (2003).
YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range
Technical contribution to the H.264 Video Coding Standard. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) Document JVT-I014r3. ITU-T & ISO/IEC MPEG Joint Video Team.
http://research.microsoft.com/apps/pubs/default.aspx?id=262595.

Martin, Sam and **Einarsson**, Per (2010).
A Real-Time Radiosity Architecture for Video Games
In: *ACM SIGGRAPH Courses*.
http://www.geomerics.com/wp-content/uploads/2014/03/radiosity%20architecture.pdf.

McCluney, William Ross (1994).
Introduction to Radiometry and Photometry.

Möller, Tomas and **Trumbore**, Ben (1997).
Fast, Minimum Storage Ray-Triangle Intersection
In: *Journal of Graphics Tools* 2.1, pp. 21–28.
DOI: 10.1080/10867651.1997.10487468.

Moreau, Pierre and **Clarberg**, Petrik (2019).
Importance Sampling of Many Lights on the GPU
In: *Ray Tracing Gems*. Ed. by Eric **Haines** and Tomas **Akenine-Möller**. 1st ed., pp. 255–283.
http://raytracinggems.com.

Müller, Thomas; **Gross**, Markus and **Novák**, Jan (2017).
Practical Path Guiding for Efficient Light-Transport Simulation
In: *Proc. of Eurographics Symposium on Rendering (EGSR)*.
DOI: 10.1111/cgf.13227.

Nichols, Greg and **Wyman**, Chris (2009).
Multiresolution Splatting for Indirect Illumination
In: *Proc. of Symposium on Interactive 3D Graphics and Games (I3D)*. I3D '09, pp. 83–90.
DOI: 10.1145/1507149.1507162.

Novák, Jan (2014).
Efficient Many-Light Rendering of Scenes with Participating Media
PhD thesis. Karlsruhe Institute of Technology.

O'Neill, Melissa E. (2014).
PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation
HMC-CS-2014-0905. Claremont, CA: Harvey Mudd College.
http://www.pcg-random.org/.

Oren, Michael and **Nayar**, Shree K. (1994).
Generalization of Lambert's Reflectance Model
In: *Proc. of Computer Graphics and Interactive Techniques (GRAPHITE)*. SIGGRAPH '94, pp. 239–246.
DOI: 10.1145/192161.192213.

Pekelis, Leonid; **Hery**, Christophe; **Villemin**, Ryusuke and **Ling**, Junyi (2015).
A Data-Driven Light Scattering Model for Hair
15-02.
https://graphics.pixar.com/library/DataDrivenHairScattering/.

Peter, Ingmar and **Pietrek**, Georg (1998).
Importance Driven Construction of Photon Maps
In: *Proc. of Eurographics Workshop on Rendering (EGWR)*, pp. 269–280.
DOI: 10.1007/978-3-7091-6453-2_25.

Pharr, Matt and **Humphreys**, Greg (2010).
Physically Based Rendering: From Theory to Implementation
2nd ed. ISBN: 0-12-375079-2 978-0-12-375079-2.

Pharr, Matt; **Jakob**, Wenzel and **Humphreys**, Greg (2017).
Physically Based Rendering: From Theory to Implementation
3rd ed.
http://www.pbrt.org/.

Polyanskiy, Mikhail N. (2018).
Refractive Index Database
https://refractiveindex.info (visited on [visited on 06/26/2018]).

Popov, Stefan; **Ramamoorthi**, Ravi; **Durand**, Fredo and **Drettakis**, George (2015).
Probabilistic Connections for Bidirectional Path Tracing
In: *Computer Graphics Forum (Proc. of EGSR)* 34.4, pp. 75–86.
DOI: 10.1111/cgf.12680.

Popoviciu, T. (1935).
Sur les équations algébriques ayant toutes leurs racines réelles
In: *Mathematica (Cluj)* 9, pp. 129–145.

Qin, Hao; **Sun**, Xin; **Hou**, Qiming; **Guo**, Baining and **Zhou**, Kun (2015).
Unbiased Photon Gathering for Light Transport Simulation
In: *ACM Transactions on Graphics (TOG)* 34.6, 208:1–208:14.
DOI: 10.1145/2816795.2818119.

Reshetov, Alexander; **Soupikov**, Alexei and **Mark**, William R. (2010).
Consistent Normal Interpolation
In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 29.6, 142:1–142:8.
DOI: 10.1145/1882261.1866168.

Rohmer, Kai; **Jendersie**, Johannes and **Grosch**, Thorsten (2017).
Natural Environment Illumination: Coherent Interactive Augmented Reality for Mobile and non-Mobile Devices
In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11, pp. 2474–2484.
DOI: 10.1109/TVCG.2017.2734426.

Rothery, P. (1982).
The Use of Control Variates in Monte Carlo Estimation of Power
In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.2, pp. 125–129.
DOI: 10.2307/2347974.
http://www.jstor.org/stable/2347974.

Rump, Martin; **Müller**, Gero; **Sarlette**, Ralf; **Koch**, Dirk and **Klein**, Reinhard (2008).
Photo-realistic Rendering of Metallic Car Paint from Image-Based Measurements
In: *Computer Graphics Forum (CGF)* 27.2. Ed. by R. **Scopigno** and E. **Gröller**, pp. 527–536.
http://cg.cs.uni-bonn.de/en/publications/paper-details/rump-2008-photo-realistic/.

Rusinkiewicz, Szymon (2004).
Estimating Curvatures and Their Derivatives on Triangle Meshes
In: *Proc. of 3D Data Processing, Visualization and Transmission*. 3DPVT, pp. 486–493.

Schjøth, Lars; **Frisvad**, Jeppe Revall; **Erleben**, Kenny and **Sporring**, Jon (2007).
Photon Differentials
In: *Proc. of Computer Graphics and Interactive Techniques (GRAPHITE)*, pp. 179–186.
DOI: 10.1145/1321261.1321293.

Schlick, Christophe (1993).
A Customizable Reflectance Model for Everyday Rendering
In: *Proc. of Eurographics Workshop on Rendering (EGWR)*, pp. 73–83.

Schregle, Roland (2003).
Bias Compensation for Photon Maps
In: *Computer Graphics Forum*.
DOI: 10.1111/j.1467-8659.2003.00720.x.

Schretter, Colas; **He**, Zhijian; **Gerber**, Mathieu; **Chopin**, Nicolas and **Niederreiter**, Harald (2016).
Van der Corput and Golden Ratio Sequences Along the Hilbert Space-Filling Curve
In: *Monte Carlo and Quasi-Monte Carlo Methods*. Ed. by Ronald **Cools** and Dirk **Nuyens**, pp. 531–544. ISBN: 978-3-319-33507-0.

Schretter, Colas; **Kobbelt**, Leif and **Dehaye**, Paul-Olivier (2012).
Golden Ratio Sequences for Low-Discrepancy Sampling
In: *Journal of Graphics Tools (JGT)* 16, pp. 95–104.

Schüssler, Vincent; **Heitz**, Eric; **Hanika**, Johannes and **Dachsbacher**, Carsten (2017).
Microfacet-based Normal Mapping for Robust Monte Carlo Path Tracing
In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 36.6, 205:1–205:12.
DOI: 10.1145/3130800.3130806.

Sheather, Simon J. (2004).
Density Estimation
In: *Statistical Science* 19.4, pp. 588–597.
http://www.jstor.org/stable/4144429.

Shirley, Peter (1991).
Physically Based Lighting Calculations for Computer Graphics
PhD thesis. University of Illinois at Urbana-Champaign.

Šik, Martin and **Křivánek**, Jaroslav (2018).
Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation
In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)*.
DOI: 10.1109/TVCG.2018.2880455.

Šik, Martin; **Otsu**, Hisanari; **Hachisuka**, Toshiya and **Křivánek**, Jaroslav (2016).
Robust Light Transport Simulation via Metropolised Bidirectional Estimators
In: *ACM Transaction on Graphics (TOG)* 35.6, 245:1–245:12.
DOI: 10.1145/2980179.2982411.

Silvennoinen, Ari and **Lehtinen**, Jaakko (2017).
Real-time Global Illumination by Precomputed Local Reconstruction from Sparse Radiance Probes
In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 36.6, 230:1–230:13.
DOI: 10.1145/3130800.3130852.
https://users.aalto.fi/~silvena4/Projects/RTGI/index.html.

Silverman, Bernard W (1986).
Density Estimation for Statistics and Data Analysis.

Smith, B. G. (1967).
Geometrical Shadowing of a Random Rough Surface
In: *IEEE Transactions on Antennas and Propagation* 15.5, pp. 668–671.
DOI: 10.1109/TAP.1967.1138991.

Stich, Martin; **Friedrich**, Heiko and **Dietrich**, Andreas (2009).
Spatial Splits in Bounding Volume Hierarchies
In: *Proc. of High Performance Graphics (HPG)*. HPG, pp. 7–13.
DOI: 10.1145/1572769.1572771.

Studios, Walt Disney Animation (2018).
Moana Island Scene
https://www.technology.disneyanimation.com/islandscene.

Surazhsky, Tatiana; **Magid**, Evgeny; **Soldea**, Octavian; **Elber**, Gershon and **Rivlin**, Ehud (2003).
A Comparison of Gaussian and Mean Curvatures Estimation Methods on Triangular Meshes
In: *IEEE International Conference on Robotics and Automation*. Vol. 1, pp. 1021–1026.
DOI: 10.1109/ROBOT.2003.1241726.

Suykens, Frank and **Willems**, Yves D. (2000).
Density Control for Photon Maps
In: *Proc. of Eurographics Workshop on Rendering (EGWR)*, pp. 23–34.
DOI: 10.1007/978-3-7091-6303-0_3.
http://graphics.cs.kuleuven.be/publications/PHOTONDC/index.html.

– (2001).
Path Differentials and Applications
In: *Proc. of Eurographics Workshop on Rendering*. EGWR, pp. 257–268.
http://dl.acm.org/citation.cfm?id=647653.732286.

Taubin, Gabriel (1995).
Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation
In: *Proc. of IEEE International Conference on Computer Vision (ICCV)*. ICCV '95, pp. 902–907.
DOI: 10.1109/ICCV.1995.466840.
http://dl.acm.org/citation.cfm?id=839277.840020.

Torrance, Kenneth E. and **Sparrow**, Ephraim M. (1967).
Theory for Off-Specular Reflection From Roughened Surfaces
In: *Journal of the Optical Society of America* 57.9, pp. 1105–1114.
DOI: 10.1364/JOSA.57.001105.

Trowbridge, T. S. and **Reitz**, K. P. (1975).
Average irregularity representation of a rough surface for ray reflection
In: *Journal of the Optical Society of America* 65.5, pp. 531–536.
DOI: 10.1364/JOSA.65.000531.

Turk, Greg (1990).
Generating Random Points in Triangles
In: *Graphics Gems*. Ed. by Andrew S. **Glassner**, pp. 24–28.
http://dl.acm.org/citation.cfm?id=90767.90772.

Veach, Eric (1997).
Robust Monte Carlo Methods for Light Transport Simulation
PhD thesis. Stanford University.
http://graphics.stanford.edu/papers/veach_thesis/.

Veach, Eric and **Guibas**, Leonidas J. (1995a).
Bidirectional Estimators for Light Transport
In: *Photorealistic Rendering Techniques*, pp. 145–167.
http://dx.doi.org/10.1007/978-3-642-87825-1_11.

– (1995b).
Optimally Combining Sampling Techniques for Monte Carlo Rendering
In: *Proceedings of SIGGRAPH '95*, pp. 419–428.
DOI: 10.1145/218380.218498.

Vorba, Jiří (2011).
Bidirectional Photon Mapping
In: *Proc. of the Central European Seminar on Computer Graphics*.
https://cgg.mff.cuni.cz/~jaroslav/papers/2011-bdpm/vorba2011-bdpm.pdf.

Vorba, Jiří and **Křivánek**, Jaroslav (2016).
Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation
In: *ACM Transactions on Graphics (TOG)* 35.4, pp. 1–11.
DOI: 10.1145/2897824.2925912.

Walter, Bruce; **Marschner**, Stephen R.; **Li**, Hongsong and **Torrance**, Kenneth E. (2007).
Microfacet Models for Refraction Through Rough Surfaces
In: *Proc. of Eurographics Symposium on Rendering (EGSR)*, pp. 195–206.
DOI: 10.2312/EGWR/EGSR07/195-206.

Wang, Zhou; **Bovik**, Alan Conrad; **Sheikh**, Hamid Rahim and **Simoncelli**, Eero P (2004).
Image Quality Assessment: from Error Visibility to Structural Similarity
In: *IEEE Transactions on Image Processing (TIP)* 13.4, pp. 600–612.
DOI: 10.1109/TIP.2003.819861.

Weber, Ernst Heinrich (1834).
De Pulsu, resorptione, auditu et tactu: Annotationes anatomicae et physiologicae...

Werner, Sebastian; **Velinov**, Zdravko; **Jakob**, Wenzel and **Hullin**, Matthias B. (2017).
Scratch Iridescence: Wave-optical Rendering of Diffractive Surface Structure
In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 36.6, 207:1–207:14.
DOI: 10.1145/3130800.3130840.

Whitted, Turner (1979).
An Improved Illumination Model for Shaded Display
In: *Computer Graphics (Proc. ACM SIGGRAPH)* 13.2, pp. 14–19.
DOI: 10.1145/965103.807419.

Wilkie, Alexander; **Nawaz**, Sehara; **Droske**, Marc; **Weidlich**, Andrea and **Hanika**, Johannes (2014).
Hero Wavelength Spectral Sampling
In: *Computer Graphics Forum (Proc. EGSR)*, pp. 123–131.
DOI: 10.1111/cgf.12419.